Red Hat Enterprise Linux 5.1

Red Hat Enterprise Linux Oracle Tuning Guide

Tuning and optimizing performance tips and guidelines for Oracle® 9i and 10g databases on Red Hat Enterprise Linux 5, 4, 3 and 2.1

Red Hat Enterprise Linux 5.1 Oracle 9i and 10g Tuning Guide
Tuning and optimizing performance tips and guidelines for Oracle® 9i and 10g databases on Red Hat Enterprise Linux 5, 4, 3 and 2.1
Edition 1.4

Editor Chris Curran ccurran@redhat.com

Copyright © 2007 by Red Hat, Inc.

Red Hat owns the copyright to this and all other subsequent derivative works. Werner Puschitz retains the copyright to the original at *www.puschitz.com*

1801 Varsity Drive
Raleigh, NC27606-2072USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588Research Triangle Park, NC27709USA

Copyright © 2007 by Red Hat, Inc. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, V1.0 or later (the latest version is presently available at *http://www.opencontent.org/openpub/*).

Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder.

Distribution of the work or derivative of the work in any standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from the copyright holder.

Red Hat and the Red Hat "Shadow Man" logo are registered trademarks of Red Hat, Inc. in the United States and other countries. Linux is a registered trademark of Linus Torvalds.

All other trademarks referenced herein are the property of their respective owners.

The GPG fingerprint of the security@redhat.com key is:

CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

This guide provides step by step instructions for tuning and optimizing Red Hat Enterprise Linux on x86 and x86-64 platforms running Oracle®9i (32 bit or 64 bit) and Oracle®10g (32 bit or 64 bit) standalone and RAC databases. It covers Red Hat Enterprise Linux Advanced Server 3 and 4 and the older version 2.1.

I. Tuning and Optimizing Red Hat Enterprise Linux for Oracle Database 9i and 10g **1**

1. Introduction **3**

2. Hardware Architectures and Linux Kernels **5**
2.1. General 5
2.2. 32 bit Architecture and the hugemem Kernel 5
2.3. The 64 bit Architecture 6

3. Kernel Upgrades **7**

4. Kernel Boot Parameters **9**
4.1. General 9
4.2. The I/O Scheduler 9

5. Memory Usage and Page Cache **11**
5.1. Checking the Memory Usage 11
5.2. Tuning the Page Cache 11

6. Swap Space **13**
6.1. General 13
6.2. Checking Swap Space Size and Usage 14

7. Setting Shared Memory **15**
7.1. Setting SHMMAX Parameter 15
7.2. Setting SHMMNI Parameter 16
7.3. Setting SHMALL Parameter 16
7.4. Removing Shared Memory 17

8. Setting Semaphores **19**
8.1. The SEMMSL Parameter 19
8.2. The SEMMNI Parameter 19
8.3. The SEMMNS Parameter 19
8.4. The SEMOPM Parameter 20
8.5. Setting Semaphore Parameters 20
8.6. An Example of Semaphore Settings 20

9. Setting File Handles **23**

10. Adjusting Network Settings **25**
10.1. Changing Network Adapter Settings 25
10.2. Changing Network Kernel Settings 25
10.3. Flow Control for e1000 Network Interface Cards 26

11. Setting Shell Limits for the Oracle User **27**
11.1. Limiting Maximum Number of Open File Descriptors for the Oracle User 27
11.2. Limiting Maximum Number of Processes Available for the Oracle User 28

12. Enabling Asynchronous I/O and Direct I/O Support **31**
12.1. Relinking Oracle9i R2 to Enable Asynchronous I/O Support 31
12.2. Relinking Oracle 10g to Enable Asynchronous I/O Support 32
12.3. Enabling Asynchronous I/O in Oracle 9i and 10g 32
12.4. Tuning Asynchronous I/O for Oracle 9i and 10g 33
12.5. Verifying Asynchronous I/O Usage 34

13. Configuring I/O for Raw Partitions **37**
13.1. General 37
13.2. Basics of Raw Devices 38

13.3. Using Raw Devices for Oracle Databases 39
13.4. Using Block Devices for Oracle 10g Release 2 in Red Hat Enterprise Linux 4 and 5 40

14. Large Memory Optimization, Big Pages, and Huge Pages 41
14.1. Big Pages in Red Hat Enterprise Linux 2.1 and Huge Pages in Red Hat Enterprise Linux 3 41
14.2. Usage of Big Pages and Huge Pages in Oracle 9i and 10g 42
14.3. Sizing Big Pages and Huge Pages 42
14.4. Checking Shared Memory Before Starting Oracle Databases 42
14.5. Configuring Big Pages in Red Hat Enterprise Linux 2.1 42
14.6. Configuring Huge Pages in Red Hat Enterprise Linux 3 43
14.7. Configuring Huge Pages in Red Hat Enterprise Linux 4 or 5 45
14.8. Huge Pages and Shared Memory File System in Red Hat Enterprise Linux 3 46

15. Growing the Oracle SGA to 2.7 GB in x86 Red Hat Enterprise Linux 2.1 Without VLM 49
15.1. General 49
15.2. Linux Memory Layout 49
15.3. Increasing Space for the SGA in Red Hat Enterprise Linux 2.1 50
15.4. Lowering the Mapped Base Address for Shared Libraries in Red Hat Enterprise Linux 2.1 51
15.5. Lowering the SGA Attach Address for Shared Memory Segments in Oracle 9i..... 51
15.6. Allowing the Oracle User to Change the Mapped Base Address for Shared Libraries 52

16. Growing the Oracle SGA to 2.7/3.42 GB in x86 Red Hat Enterprise Linux 3, 4 and 5 Without VLM 55
16.1. General 55
16.2. Mapped Base Address for Shared Libraries in Red Hat Enterprise Linux 3, 4 and 5 55
16.3. Oracle 10g SGA Sizes in Red Hat Enterprise Linux 3, 4 or 5 56
16.4. Lowering the SGA Attach Address in Oracle 10g 57

17. Using Very Large Memory (VLM) 59
17.1. General 59
17.2. Configuring Very Large Memory (VLM) 60

II. Installing the Oracle Database 10g on Red Hat Enterprise Linux 63

18. Downloading and Unpacking Oracle 10g Installation Files 65

19. Pre-Installation Preparation and Verification 67
19.1. Verifying Memory and Swap Space 67
19.2. Verifying Temporary(`/tmp`) Space 67
19.3. Verifying Software Packages (RPMs) 67
19.4. Verifying Kernel Parameters 71

20. Installing Required Software Packages 75
20.1. 10g R2 on Red Hat Enterprise Linux 4 and 5 x86-64 version 75
20.2. 10g R2 on Red Hat Enterprise Linux 4 and 5 (x86) 77
20.3. 10g R1 on Red Hat Enterprise Linux 4 and 5 (x86-64) 78
20.4. 10g R1 on Red Hat Enterprise Linux 4 and 5 (x86) 78
20.5. Oracle 10g R1 and R2 on Red Hat Enterprise Linux 3 (x86) 79
20.6. Oracle 10g R1 on Red Hat Enterprise Linux 3 (x86_64) 79

20.7. Oracle 10g R1 on Red Hat Enterprise Linux 2.1 (x86) 80
20.8. Verifying and Updating the redhat-release File 80

21. Sizing Disk Space for Oracle 10g **83**

22. Setting Shell Limits for Your Oracle User **85**

23. Creating Oracle User Accounts **87**

24. Creating Oracle Directories **89**
24.1. Optimal Flexible Architecture (OFA) for 10g R1 (10.1.0.2) 89

25. Setting Oracle Environments **91**

26. Installing Oracle Database 10g **93**
26.1. Installing Oracle 10g on a Remote Linux Server 93
26.2. Installing Oracle 10gR2 Cluster Ready Services (CRS) with MPIO 93
26.3. Starting Oracle Universal Installer 97
26.4. Using Oracle Universal Installer (OUI) 98
26.5. Updating after the Oracle Universal Installer 100

27. Oracle Post Installation Tasks **103**
27.1. Startup and Shutdown of the Oracle 10g Database 103
27.2. Shutdown of other Oracle 10g Background Processes 103

28. Tips and Hints for Oracle 10g on Red Hat Enterprise Linux **105**

29. Oracle 10g and Linux Installation Errors and Troubleshooting **107**

III. Installing the Oracle9i 32 bit Database on Red Hat Enterprise Linux **115**

30. Preparing Red Hat Enterprise Linux for an Oracle Database Installation **117**
30.1. Unpacking and Downloading the Oracle9i Installation Files 117
30.2. Setting Swap Space 117
30.3. Setting Shared Memory 118
30.4. Examining Temporary(/tmp) Space 118
30.5. Sizing Oracle Disk Space 118

31. Verifying Required Packages(RPMs) **119**
31.1. Required Packages for Red Hat Advanced Server 2.1 119
31.2. Required Packages for Red Hat Enterprise Linux 3 119
31.3. Required Packages for Red Hat Enterprise Linux 4 120

32. Setting Up a Working Environment for Oracle **123**
32.1. Creating Oracle User Accounts 123
32.2. Creating Oracle Directories 123
32.3. Setting Oracle Environment Variables 123

33. Starting runInstaller **125**

34. Installing Oracle9i R2 (9.2.0.1.0) on Red Hat Advanced Server 2.1 **127**

35. Installing Oracle9i R2 (9.2.0.4.0) on Red Hat Enterprise Linux 3 **129**
35.1. Installing Oracle9i R2 (9.2.0.1.0) on Red Hat Enterprise Linux 3 129
35.2. Patching Oracle9i to 9.2.0.4.0 on Red Hat Enterprise Linux 3 132
35.3. Patching Oracle Intelligent Agent on Red Hat Enterprise Linux 3 134

36. Installing Oracle9i R2 (9.2.0.6.0) on Red Hat Enterprise Linux 4 **137**
36.1. Installing Oracle9i R2 (9.2.0.4.0) on Red Hat Enterprise Linux 4 137

36.2. Patching Oracle9i R2 to 9.2.0.6.0 on Red Hat Enterprise Linux 4 138

37. Starting and Shutting down the Oracle9i Database **141**

38. Oracle Installation Errors **143**

39. Reference List **151**

A. Revision History **153**

Part I. Tuning and Optimizing Red Hat Enterprise Linux for Oracle Database 9i and 10g

Introduction

This guide covers optimizations for Red Hat Enterprise Linux x86 (32 bit) and x86_64 (64 bit) platforms running Oracle 9i R2 (32 bit or 64 bit) and Oracle 10g R1/R2 (32 bit or 64 bit) standalone and RAC databases. This white paper covers Red Hat Enterprise Linux Advanced Server 2.1, 3, 4 and 5. Various workarounds covered in this article are due to the 32 bit address limitations of the x86 platform. However, many steps described in this document also apply to x86-64 platforms. Sections that do not specifically say that its only applicable to 32bit or 64bit apply to both platforms. If you think that a section is not very clear on that, let me know.For supported system configurations and limits for Red Hat Enterprise Linux releases, see *http://www.redhat.com/rhel/details/limits/*. For instructions on installing Oracle 10g and 9i databases on Red Hat Enterprise Linux, see Appendix A and Appendix B respectively.

Hardware Architectures and Linux Kernels

2.1. General

When it comes to large databases the hybrid x86-64 architecture platform is strongly recommended over the 32 bit x86 platform. 64 bit platforms can access more than 4GB of memory without workarounds. With 32 bit platforms there are several issues that require workaround solutions for databases that use lots of memory, for example refer to *Chapter 17, Using Very Large Memory (VLM)*. If you are not sure whether you are on a 32 bit or 64 bit hardware, run **`dmidecode`** or **`cat /proc/cpuinfo`**. Running **`uname -a`** can be misleading since 32 bit Linux kernels can run on x86-64 platforms. But if **`uname -a`** displays x86-64, then you are running a 64 bit Linux kernel on a x86_64 platform.

2.2. 32 bit Architecture and the hugemem Kernel

In Red Hat Enterprise Linux 3, 4 or 5 the smp kernel can be used on systems with up to 16 GB of RAM. The hugemem kernel is required in order to use all the memory on systems that have more than 16GB of RAM up to 64GB. *However, it is recommend to use the hugemem kernel even on systems that have 8GB of RAM or more due to the potential issue of "low memory" starvation (see next section) that can happen on database systems with 8 GB of RAM*. The stability you get with the hugemem kernel on larger systems outperforms the performance overhead of address space switching.

With x86 architecture the first 16MB-896MB of physical memory is known as "low memory" (ZONE_NORMAL) which is permanently mapped into kernel space. Many kernel resources must live in the low memory zone. In fact, many kernel operations can only take place in this zone. This means that the low memory area is the most performance critical zone. For example, if you run resource-intensive applications or have a lot of memory installed and are simultaneously running resource-intensive applications, memory addresses below 896MB can become constrained. This happens because more kernel structures must be allocated to these low memory addresses. Low memory starvation happens when **`LowFree`** in **`/proc/meminfo`** becomes very low accompanied by a sudden spike in paging activity. To free up memory in the low memory zone, the kernel bounces buffers aggressively between low memory and high memory which becomes noticeable as paging (do not confuse it with paging to the swap partition). If the kernel is unable to free up enough memory in the low memory zone, then the kernel can hang the system.

Paging activity can be monitored using the **`vmstat`** command or using the **`sar`** command (option '**`-B`**') which comes with the **`sysstat`** RPM. Since Linux tries to utilize the whole low memory zone, a low LowFree in **`/proc/meminfo`** does not necessarily mean that the system is out of low memory. However, when the system shows increased paging activity when **`LowFree`** gets below 50MB, then the hugemem kernel should be installed. The stability you gain from using the hugemem kernel makes up for any performance impact resulting from the 4GB-4GB kernel/user memory split in this kernel (a classic 32 bit x86 system splits the available 4 GB address space into 3 GB virtual memory space for user processes and a 1 GB space for the kernel). To see some allocations in the low memory zone, refer to **`/proc/meminfo`** and **`slabtop(1)`** for more information. Note that Huge Pages would free up memory in the low memory zone since the system has less bookkeeping to do for that part of virtual memory, see *Chapter 17, Using Very Large Memory (VLM)*.

If you install the Red Hat Enterprise Linux 3, 4 or 5 hugemem kernel ensure that any proprietary drivers you are using (e.g. proprietary multipath drivers) are certified with the hugemem kernel.

In Red Hat Enterprise Linux 2.1, the smp kernel is capable of handling up to 4GB of RAM. The **`kernel-enterprise`** kernel should be used for systems with more than 4GB of RAM up to 16GB.

In Red Hat Enterprise Linux 5, a 32 bit kernel is always a hugemem kernel so there is no need to install a special kernel.

2.3. The 64 bit Architecture

This is the architecture that should be used whenever possible. If you can go with a x86_64 platform ensure that all drivers you need are supported on x86_64 (for example proprietary multipath drivers) Furthermore, ensure that all the required applications are supported on x86_64 as well.

Kernel Upgrades

Make sure to install the latest kernel where all proprietary drivers, if applicable, are certified and supported. Note that proprietary drivers are often installed under **`/lib/modules/<kernel-version>/kernel/drivers/addon`**. For example, the **EMC PowerPath** drivers can be found in the following directory when running the **`2.4.21-32.0.1.ELhugemem`** kernel:

```
$ ls -al /lib/modules/2.4.21-32.0.1.ELhugemem/kernel/drivers/addon/emcpower
total 732
drwxr-xr-x    2 root     root         4096 Aug 20 13:50 .
drwxr-xr-x   19 root     root         4096 Aug 20 13:50 ..
-rw-r--r--    1 root     root        14179 Aug 20 13:50 emcphr.o
-rw-r--r--    1 root     root         2033 Aug 20 13:50 emcpioc.o
-rw-r--r--    1 root     root        91909 Aug 20 13:50 emcpmpaa.o
-rw-r--r--    1 root     root       131283 Aug 20 13:50 emcpmpap.o
-rw-r--r--    1 root     root       113922 Aug 20 13:50 emcpmpc.o
-rw-r--r--    1 root     root        75380 Aug 20 13:50 emcpmp.o
-rw-r--r--    1 root     root       263243 Aug 20 13:50 emcp.o
-rw-r--r--    1 root     root         8294 Aug 20 13:50 emcpsf.o
$
```

Therefore, when you upgrade the kernel you must ensure that all proprietary modules can be found in the right directory so that the kernel can load them. To check which kernels are installed, run the following command:

```
$ rpm -qa | grep kernel
```

To check which kernel is currently running, execute the following command:

```
$ uname -r
```

For example, to install the 2.4.21-32.0.1.ELhugemem kernel, download the kernel-hugemem RPM and execute the following command:

```
# rpm -ivh kernel-hugemem-2.4.21-32.0.1.EL.i686.rpm
```

Note

Never upgrade the kernel using the RPM option '**`-U`**'. The previous kernel should always be available if the newer kernel does not boot or work properly.

To make sure the right kernel is booted, check the **`/etc/grub.conf`** file if you use **GRUB** and change the "**`default`**" attribute if necessary. Here is an example:

```
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
title Red Hat Enterprise Linux AS (2.4.21-32.0.1.ELhugemem)
 root (hd0,0)
 kernel /vmlinuz-2.4.21-32.0.1.ELhugemem ro root=/dev/sda2
 initrd /initrd-2.4.21-32.0.1.ELhugemem.img
title Red Hat Enterprise Linux AS (2.4.21-32.0.1.ELsmp)
 root (hd0,0)
 kernel /vmlinuz-2.4.21-32.0.1.ELsmp ro root=/dev/sda2
```

```
initrd /initrd-2.4.21-32.0.1.ELsmp.img
```

In this example, the "**default**" attribute is set to "**0**" which means that the **2.4.21-32.0.1.ELhugemem kernel** will be booted. If the "**default**" attribute would be set to "**1**", then **2.4.21-32.0.1.ELsmp** would be booted. After you installed the newer kernel reboot the system. Once you are sure that you do not need the old kernel anymore, you can remove the old kernel by running:

```
# rpm -e <OldKernelVersion>
```

When you remove a kernel, you do not need to update **/etc/grub.conf**.

Kernel Boot Parameters

4.1. General

The Linux kernel accepts boot parameters when the kernel is started. Very often it is used to provide information to the kernel about hardware parameters where the kernel would have problems or to overwrite default values. Red Hat Enterprise Linux 3 uses 2.4 based kernels, Red Hat Enterprise Linux 4 and 5 use 2.6 based kernels.

For a list of kernel parameters in Red Hat Enterprise Linux 4 and 5, see **`/usr/share/doc/kernel-doc-2.6.9/Documentation/kernel-parameters.txt`**. This file does not exist if the **`kernel-doc`** RPM is not installed. And for a list of kernel parameters in Red Hat Enterprise Linux3 and Red Hat Enterprise Linux 2.1, see **`/usr/src/linux-2.4/Documentation/kernel-parameters.txt`** which comes with the **`kernel-doc`** RPM.

4.2. The I/O Scheduler

Starting with the 2.6 kernel, for example Red Hat Enterprise Linux 4 or 5, the I/O scheduler can be changed at boot time which controls the way the kernel commits reads and writes to disks. For more information on various I/O scheduler, see *Choosing an I/O Scheduler for Red Hat Enterprise Linux 4 and the 2.6 Kernel* [1]. Red Hat Enterprise Linux 5 allows users to change I/O schedulers dynamically, one way this can be done is by executing the command **`echo sched_name > /sys/block/<sdx>/queue/scheduler`**.

The *Completely Fair Queuing* (CFQ) scheduler is the default algorithm in Red Hat Enterprise Linux 4 which is suitable for a wide variety of applications and provides a good compromise between throughput and latency. In comparison to the CFQ algorithm, the *Deadline* scheduler caps maximum latency per request and maintains a good disk throughput which is best for disk-intensive database applications. Hence, the *Deadline* scheduler is recommended for database systems. Also, at the time of this writing there is a bug in the CFQ scheduler which affects heavy I/O, see Metalink Bug:5041764. Even though this bug report talks about OCFS2 testing, this bug can also happen during heavy IO access to raw or block devices and as a consequence could evict RAC nodes.

To switch to the Deadline scheduler, the boot parameter **`elevator=deadline`** must be passed to the kernel that is being used. Edit the **`/etc/grub.conf`** file and add the following parameter to the kernel that is being used, in this example **`2.4.21-32.0.1.ELhugemem`**:

```
title Red Hat Enterprise Linux Server (2.6.18-8.el5)
 root (hd0,0)
 kernel /vmlinuz-2.6.18-8.el5 ro root=/dev/sda2 elevator=deadline
 initrd /initrd-2.6.18-8.el5.img
```

This entry tells the **`2.6.18-8.el5`** kernel to use the *Deadline* scheduler. Make sure to reboot the system to activate the new scheduler.

[1] http://www.redhat.com/magazine/008jun05/features/schedulers/

Memory Usage and Page Cache

5.1. Checking the Memory Usage

To determine the size and usage of memory, you can enter the following command:

```
grep MemTotal /proc/meminfo
```

You can find a detailed description of the entries in **/proc/meminfo** at *http://www.redhat.com/advice/tips/meminfo.html*.

Alternatively, you can use the **free(1)** command to check the memory:

```
$ free
             total       used       free     shared    buffers     cached
Mem:       4040360    4012200      28160          0     176628    3571348
-/+ buffers/cache:     264224    3776136
Swap:      4200956      12184    4188772
$
```

In this example the total amount of available memory is 4040360 KB. 264224 KB are used by processes and 3776136 KB are free for other applications. Do not get confused by the first line which shows that 28160KB are free! If you look at the usage figures you can see that most of the memory use is for buffers and cache. Linux always tries to use RAM to speed up disk operations by using available memory for buffers (file system metadata) and cache (pages with actual contents of files or block devices). This helps the system to run faster because disk information is already in memory which saves I/O operations. If space is needed by programs or applications like Oracle, then Linux will free up the buffers and cache to yield memory for the applications. If your system runs for a while you will usually see a small number under the field "**free**" on the first line.

5.2. Tuning the Page Cache

Page Cache is a disk cache which holds data of files and executable programs, for example pages with actual contents of files or block devices. Page Cache (disk cache) is used to reduce the number of disk reads. To control the percentage of total memory used for page cache in Red Hat Enterprise Linux 3, the following kernel parameter can be changed:

```
# cat /proc/sys/vm/pagecache
1       15      30
```

The above three values are usually good for database systems. It is not recommended to set the third value very high, around 100, as it used to be with older Red Hat Enterprise Linux 3 kernels. This can cause significant performance problems for database systems. If you upgrade to a newer kernel like **2.4.21-37**, then these values will automatically change to "**1 15 30**" unless its set to different values in **/etc/sysctl.conf**. For information on tuning the **pagecache** kernel parameter, see Understanding Virtual Memory. Note this kernel parameter does not exist in Red Hat Enterprise Linux 4.

The *pagecache* parameters can be changed in the proc file system without reboot:

```
# echo "1 15 30" > /proc/sys/vm/pagecache
```

Alternatively, you can use **sysctl(8)** to change it:

```
# sysctl -w vm.pagecache="1 15 30"
```

To make the change permanent, add the following line to the file **/etc/sysctl.conf**. This file is used during the boot process.

```
# echo "vm.pagecache=1 15 30" >> /etc/sysctl.conf
```

With Red Hat Enterprise Linux 4 and 5, the *pagecache* is dynamically adjusted. You can adjust the minimum free pages using the following command:

```
# echo 1024 > /proc/sys/vm/min_free_kbytes
```

To make the change permanent, add the following line to the file **/etc/sysctl.conf**:

```
# echo vm.min_free_kbytes=1024 >> /etc/sysctl.conf
```

The **pagecache** pages can be reclaimed by adjusting the *swappiness* percentage as described in the next section.

Swap Space

6.1. General

In some cases it is good for the swap partition to be used. For example, long running processes often access only a subset of the page frames they obtained. This means that the swap partition can safely be used even if memory is available because system memory could be better served for disk cache to improve overall system performance. In fact, in the 2.6 kernel used in Red Hat Enterprise Linux 4 and 5, you can define a threshold when processes should be swapped out in favor of I/O caching. This can be tuned with the **/proc/sys/vm/swappiness** kernel parameter. The default value of **/proc/sys/vm/swappiness** is 60 which means that applications and programs that have not done a lot lately can be swapped out. Higher values will provide more I/O cache and lower values will wait longer to swap out idle applications. **Swappiness** percentage may be tuned using:

```
# echo 10 > /proc/sys/vm/swappiness
```

or,

```
# echo vm.swappiness=10 >> /etc/sysctl.conf
```

Depending on your system profile you may see that swap usage slowly increases with the time the system is up. To display swap usage you can run the **free(1)** command or you can check the **/proc/meminfo** file. When the system uses swap space it will sometimes not decrease afterward. This saves I/O if memory is needed and pages do not have to be swapped out again when the pages are already in the swap space. However, if swap usage gets close to 80% - 100% (your threshold may be lower if you use a large swap space), then a closer look should be taken at the system, see also *Section 6.2, "Checking Swap Space Size and Usage"*. Depending on the size of your swap space, you may want to check swap activity with **vmstat** or **sar** if swap allocation is lower than 80%. But these numbers really depend on the size of the swap space. The **vmstat** or **sar** command output the number of pages swapped. This output field is an important metric. This number should be low or zero as constant page swapping should be avoided at all costs.

Note

Never add a permanent swap file to the system due to the performance impact on the file system layer.

Swap Size Recommendations According to Oracle9i Installation Guide Release 2 a minimum of 512MB of RAM is required to install Oracle9i Server. According to Oracle Database Installation Guide 10g Release 2 at least 1024MB of RAM is required for 10g R2.

For 10g R2, Oracle gives the following swap space requirement:

RAM	Swap Space
1 GB - 2 GB	1.5 times the size of RAM
2 GB - 8 GB	Equal to the size of RAM
Greater than 8GB	0.75 times the size of RAM

Table 6.1. Recommended Swap Space Requirements for 10g R2

6.2. Checking Swap Space Size and Usage

You can check the size and current usage of swap space by running the command: **`grep SwapTotal /proc/meminfo`**

Swap usage may slowly increase as shown above but should stop at some point. If swap usage continues to grow steadily or is already large, then one of the following choices may need to be considered:

- Adding more RAM.
- Reducing the size of the SGA.
- Increasing the size of the swap space.

If you see constant swapping, then you need to either add more RAM or reduce the size of the SGA. Constant swapping should be avoided at all cost. You can check current swap activity using the following commands:

```
$ vmstat 3 100
procs                  memory    swap         io     system         cpu
 r  b swpd   free  buff cache si   so   bi    bo   in    cs us sy id wa
 1  0    0 972488  7148 20848  0    0  856     6  138    53  0  0 99  0
 0  1    0 962204  9388 20848  0    0  747     0 4389  8859 23 24 11 41
 0  1    0 959500 10728 20848  0    0  440   313 1496  2345  4  7  0 89
 0  1    0 956912 12216 20848  0    0  496     0 2294  4224 10 13  0 77
 1  1    0 951600 15228 20848  0    0  997   264 2241  3945  6 13  0 81
 0  1    0 947860 17188 20848  0    0  647   280 2386  3985  9  9  1 80
 0  1    0 944932 19304 20848  0    0  705     0 1501  2580  4  9  0 87
```

The fields si and so show the amount of memory paged in from disk and paged out to disk, respectively. If the server shows continuous swap activity then more memory should be added or the SGA size should be reduced. To check the history of swap activity, you can use the sar command. For example, to check swap activity from Oct 12th:

```
# ls -al /var/log/sa | grep "Oct 12"
-rw-r--r--    1 root     root      2333308 Oct 12 23:55 sa12
-rw-r--r--    1 root     root      4354749 Oct 12 23:53 sar12
# sar -W -f /var/log/sa/sa12
Linux 2.4.21-32.0.1.ELhugemem (rac01prd)       10/12/2005

12:00:00 AM  pswpin/s pswpout/s
12:05:00 AM      0.00      0.00
12:10:00 AM      0.00      0.00
12:15:00 AM      0.00      0.00
12:20:00 AM      0.00      0.00
12:25:00 AM      0.00      0.00
12:30:00 AM      0.00      0.00
...
```

The fields **`pswpin`** and **`pswpout`** show the total number of pages brought in and out per second, respectively.

If the server shows sporadic swap activity or swap activity for a short period time at certain intervals, then you can either add more swap space or RAM. If swap usage is already very large, do not confuse very large swap usage with constant swapping, then more RAM is recommended.

Setting Shared Memory

Shared memory allows processes to access common structures and data by placing them in shared memory segments. It is the fastest form of inter-process communication available since no kernel involvement occurs when data is passed between the processes. In fact, data does not need to be copied between the processes.

Oracle uses shared memory segments for the Shared Global Area (SGA) which is an area of memory that is shared by Oracle processes. The size of the SGA has a significant impact to Oracle's performance since it holds database buffer cache and much more.

To see all shared memory settings, execute:

```
$ ipcs -lm
```

7.1. Setting SHMMAX Parameter

This parameter defines the maximum size in bytes of a single shared memory segment that a Linux process can allocate in its virtual address space. For example, if you use the Red Hat Enterprise Linux 3 **smp** kernel on a 32 bit platform (x86), then the virtual address space for a user process is 3 GB. If you use the Red Hat Enterprise Linux 3 **hugemem** kernel on a 32 bit platform (x86), then the virtual address space for a user process is almost 4GB. Hence, setting **SHMMAX** to 4GB - 1 byte (4294967295 bytes) on a smp kernel on a 32 bit architecture will not increase the maximum size of a shared memory segment to 4 GB -1. Even setting **SHMMAX** to 4 GB - 1 byte using the **hugemem** kernel on a 32 bit architecture will not enable a process to get such a large shared memory segment. In fact, the upper limit for a shared memory segment for an Oracle 10g R1 SGA using the **hugemem** kernel is roughly 3.42 GB (~3.67 billion bytes) since virtual address space is also needed for other things like shared libraries. This means if you have three 2 GB shared memory segments on a 32 bit system, no process can attach to more than one shared memory segment at a time. Also note if you set **SHMMAX** to 4294967296 bytes (4*1024*1024*1024=4GB) on a 32 bit system, then **SHMMAX** will essentially bet set to 0 bytes since it wraps around the 4GB value. This means that **SHMMAX** should not exceed 4294967295 on a 32 bit system. On x86-64 platforms, **SHMMAX** can be much larger than 4GB since the virtual address space is not limited by 32 bits.

Since the SGA is comprised of shared memory, **SHMMAX** can potentially limit the size of the SGA. **SHMMAX** should be slightly larger than the SGA size. If **SHMMAX** is too small, you can get error messages similar to this one:

```
ORA-27123: unable to attach to shared memory segment
```

It is highly recommended that the shared memory fits into the Big Pages or Huge Pages pool, see *Chapter 14, Large Memory Optimization, Big Pages, and Huge Pages*.

To increase the default maximum SGA size on x86 Red Hat Enterprise Linux 2.1 systems without VLM, refer to *Chapter 15, Growing the Oracle SGA to 2.7 GB in x86 Red Hat Enterprise Linux 2.1 Without VLM*.

To increase the default maximum SGA size on x86 Red Hat Enterprise Linux 3, 4 and 5 systems without VLM, refer to *Chapter 16, Growing the Oracle SGA to 2.7/3.42 GB in x86 Red Hat Enterprise Linux 3, 4 and 5 Without VLM*..

To determine the maximum size of a shared memory segment, run:

```
# cat /proc/sys/kernel/shmmax
2147483648
```

The default shared memory limit for SHMMAX can be changed in the **proc** file system without reboot:

```
# echo 2147483648 > /proc/sys/kernel/shmmax
```

Alternatively, you can use **sysctl(8)** to change it:

```
# sysctl -w kernel.shmmax=2147483648
```

To make a change permanent, add the following line to the file **/etc/sysctl.conf** (your setting may vary). This file is used during the boot process.

```
# echo "kernel.shmmax=2147483648" >> /etc/sysctl.conf
```

7.2. Setting SHMMNI Parameter

This parameter sets the system wide maximum number of shared memory segments.

Oracle recommends SHMMNI to be at least 4096 for Oracle 10g. For Oracle 9i on x86 the recommended minimum setting is lower. Since these recommendations are minimum settings, it is best to set it always to at least 4096 for 9i and 10g databases on x86 and x86-64 platforms.

To determine the system wide maximum number of shared memory segments, run:

```
# cat /proc/sys/kernel/shmmni
4096
```

The default shared memory limit for SHMMNI can be changed in the **proc** file system without reboot:

```
# echo 4096 > /proc/sys/kernel/shmmni
```

Alternatively, you can use **sysctl(8)** to change it:

```
# sysctl -w kernel.shmmni=4096
```

To make a change permanent, add the following line to the file **/etc/sysctl.conf**. This file is used during the boot process.

```
# echo "kernel.shmmni=4096" >> /etc/sysctl.conf
```

7.3. Setting SHMALL Parameter

This parameter sets the total amount of shared memory pages that can be used system wide. Hence, SHMALL should always be at least **ceil(shmmax/PAGE_SIZE)**.

The default size for SHMALL in Red Hat Enterprise Linux 2.1, 3, 4 and 5 is 2097152 which is also Oracle's recommended minimum setting for 9i and 10g on x86 and x86-64 platforms. In most cases this setting should be sufficient since it means that the total amount of shared memory available on the

system is 2097152*4096 bytes **(shmall*PAGE_SIZE)** which is 8 GB. **PAGE_SIZE** is usually 4096 bytes unless you use *Chapter 14, Large Memory Optimization, Big Pages, and Huge Pages* which supports the configuration of larger memory pages.

If you are not sure what the default **PAGE_SIZE** is on your Linux system, you can run the following command:

```
$ getconf PAGE_SIZE
4096
```

To determine the system wide maximum number of shared memory pages, run:

```
# cat /proc/sys/kernel/shmall
2097152
```

The default shared memory limit for SHMALL can be changed in the **proc** file system without reboot:

```
# echo 2097152 > /proc/sys/kernel/shmall
```

Alternatively, you can use **sysctl(8)** to change it:

```
# sysctl -w kernel.shmall=2097152
```

To make the change permanent, add the following line to the file **/etc/sysctl.conf**. This file is used during the boot process.

```
# echo "kernel.shmall=2097152" >> /etc/sysctl.conf
```

7.4. Removing Shared Memory

Sometimes after an instance crash you may have to remove Oracle's shared memory segments manually.

To see all shared memory segments that are allocated on the system, execute:

```
$ ipcs -m

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x8f6e2129 98305      oracle     600        77694523   0
0x2f629238 65536      oracle     640        2736783360 35
0x00000000 32768      oracle     640        2736783360 0           dest
```

In this example you can see that three shared memory segments have been allocated. The output also shows that shmid 32768 is an abandoned shared memory segment from a past ungraceful Oracle shutdown. Status "**dest**" means that this memory segment is marked to be destroyed. To find out more about this shared memory segment you can run:

```
$ ipcs -m -i 32768
Shared memory Segment shmid=32768
uid=500 gid=501 cuid=500 cgid=501
mode=0640 access_perms=0640
```

```
bytes=2736783360 lpid=3688 cpid=3652 nattch=0
att_time=Sat Oct 29 13:36:52 2005
det_time=Sat Oct 29 13:36:52 2005
change_time=Sat Oct 29 11:21:06 2005
```

To remove the shared memory segment, you could copy and paste **shmid** and execute:

```
$ ipcrm shm 32768
```

Another approach to remove shared memory is to use Oracle's **sysresv** utility. Here are a few self explanatory examples on how to use **sysresv**:

Checking Oracle's IPC resources:

```
$ sysresv

IPC Resources for ORACLE_SID "orcl" :
Shared Memory
ID              KEY
No shared memory segments used
Semaphores:
ID              KEY
No semaphore resources used
Oracle Instance not alive for sid "orcl"
$
```

Instance is up and running:

```
$ sysresv -i

IPC Resources for ORACLE_SID "orcl" :
Shared Memory:
ID              KEY
2818058         0xdc70f4e4
Semaphores:
ID              KEY
688128          0xb11a5934
Oracle Instance alive for sid "orcl"
SYSRESV-005: Warning
Instance maybe alive - aborting remove for sid "orcl"
$
```

Instance has crashed and resources were not released:

```
$ sysresv -i

IPC Resources for ORACLE_SID "orcl" :
Shared Memory:
ID              KEY
32768           0xdc70f4e4
Semaphores:
ID              KEY
98304           0xb11a5934
Oracle Instance not alive for sid "orcl"
Remove ipc resources for sid "orcl" (y/n)?y
Done removing ipc resources for sid "orcl"
$
```

Setting Semaphores

Semaphores can best be described as counters which are used to provide synchronization between processes or between threads within a process for shared resources like shared memories. **System V** semaphores support semaphore sets where each one is a counting semaphore. So when an application requests semaphores, the kernel releases them in sets. The number of semaphores per set can be defined through the kernel parameter SEMMSL.

To see all semaphore settings, run:

```
ipcs -ls
```

8.1. The SEMMSL Parameter

This parameter defines the maximum number of semaphores per semaphore set.

Oracle recommends SEMMSL to be at least 250 for 9i R2 and 10g R1/R2 databases except for 9i R2 on x86 platforms where the minimum value is lower. Since these recommendations are minimum settings, it is best to set it always to at least 250 for 9i and 10g databases on x86 and x86-64 platforms.

Note

If a database gets thousands of concurrent connections where the **`ora.init`** parameter **PROCESSES** is very large, then SEMMSL should be larger as well. Note what **Metalink** Note:187405.1 and Note:184821.1 have to say regarding SEMMSL: "The SEMMSL setting should be 10 plus the largest **PROCESSES** parameter of any Oracle database on the system". Even though these notes talk about 9i databases this SEMMSL rule also applies to 10g databases. I have seen low SEMMSL settings to be an issue for 10g RAC databases where Oracle recommended to increase SEMMSL and to calculate it according to the rule mentioned in these notes. An example for setting semaphores for higher **PROCESSES** settings can be found at *Section 8.6, "An Example of Semaphore Settings"*.

8.2. The SEMMNI Parameter

This parameter defines the maximum number of semaphore sets for the entire Linux system.

Oracle recommends SEMMNI to be at least 128 for 9i R2 and 10g R1/R2 databases except for 9i R2 on x86 platforms where the minimum value is lower. Since these recommendations are minimum settings, it is best to set it always to at least 128 for 9i and 10g databases on x86 and x86-64 platforms.

8.3. The SEMMNS Parameter

This parameter defines the total number of semaphores (not semaphore sets) for the entire Linux system. A semaphore set can have more than one semaphore, and as the **`semget(2) man`** page explains, values greater than SEMMSL * SEMMNI makes it irrelevant. The maximum number of semaphores that can be allocated on a Linux system will be the lesser of: SEMMNS or (SEMMSL * SEMMNI).

Oracle recommends SEMMSL to be at least 32000 for 9i R2 and 10g R1/R2 databases except for 9i R2 on x86 platforms where the minimum value is lower. Setting SEMMNS to 32000 ensures that SEMMSL * SEMMNI (250*128=32000) semaphores can be be used. Therefore it is recommended to set SEMMNS to at least 32000 for 9i and 10g databases on x86 and x86-64 platforms.

8.4. The SEMOPM Parameter

This parameter defines the maximum number of semaphore operations that can be performed per **`semop(2)`** system call (semaphore call). The **`semop(2)`** function provides the ability to do operations for multiple semaphores with one **`semop(2)`** system call. Since a semaphore set can have the maximum number of SEMMSL semaphores per semaphore set, it is often recommended to set SEMOPM equal to SEMMSL.

Oracle recommends to set SEMOPM to a minimum value of 100 for 9i R2 and 10g R1/R2 databases on x86 and x86-64 platforms.

8.5. Setting Semaphore Parameters

To determine the values of the four described semaphore parameters, run:

```
# cat /proc/sys/kernel/sem
250     32000   32      128
```

These values represent SEMMSL, SEMMNS, SEMOPM, and SEMMNI.

Alternatively, you can run:

```
# ipcs -ls
```

All four described semaphore parameters can be changed in the proc file system without reboot:

```
# echo 250 32000 100 128 > /proc/sys/kernel/sem
```

Alternatively, you can use sysctl(8) to change it:

```
sysctl -w kernel.sem="250 32000 100 128"
```

To make the change permanent, add or change the following line in the file **`/etc/sysctl.conf`**. This file is used during the boot process.

```
echo "kernel.sem=250 32000 100 128" >> /etc/sysctl.conf
```

8.6. An Example of Semaphore Settings

On systems where the **`ora.init`** parameter **PROCESSES** is very large, the semaphore settings need to be adjusted accordingly.

As shown at *Section 24.1, “Optimal Flexible Architecture (OFA) for 10g R1 (10.1.0.2)”* the SEMMSL setting should be 10 plus the largest **PROCESSES** parameter of any Oracle database on the system. So if you have one database instance running on a system where **PROCESSES** is set to 5000, then SEMMSL should be set to 5010.

As shown at *Section 8.3, "The SEMMNS Parameter"* the maximum number of semaphores that can be allocated on a Linux system will be the lesser of: SEMMNS or (SEMMSL * SEMMNI). Since SEMMNI can stay at 128, we need to increase SEMMNS to 641280 (5010*128).

As shown at *Section 8.4, "The SEMOPM Parameter"* a semaphore set can have the maximum number of SEMMSL semaphores per semaphore set and it is recommended to set SEMOPM equal to SEMMSL. Since SEMMSL is set to 5010 the SEMOPM parameter should be set to 5010 as well.

Hence, if the **`ora.init`** parameter **PROCESSES** is set to 5000, then the semaphore settings should be as follows:

```
sysctl -w kernel.sem="5010 641280 5010 128"
```

Setting File Handles

The maximum number of file handles denotes the maximum number of open files on a Linux system.

Oracle recommends that the file handles for the entire system is set to at least 65536 for 9i R2 and 10g R1 and R2 for x86 and x86-64 platforms.

To determine the maximum number of file handles for the entire system, run:

```
cat /proc/sys/fs/file-max
```

To determine the current usage of file handles, run:

```
$ cat /proc/sys/fs/file-nr
1154    133     8192
```

The **`file-nr`** file displays three parameters:

- the total allocated file handles.
- the number of currently used file handles (with the 2.4 kernel); or the number of currently unused file handles (with the 2.6 kernel).
- the maximum file handles that can be allocated (also found in /proc/sys/fs/file-max).

The kernel dynamically allocates file handles whenever a file handle is requested by an application but the kernel does not free these file handles when they are released by the application. The kernel recycles these file handles instead. This means that over time the total number of allocated file handles will increase even though the number of currently used file handles may be low.

The maximum number of file handles can be changed in the **`proc`** file system without reboot:

```
# echo 65536 > /proc/sys/fs/file-max
```

Alternatively, you can use **`sysctl(8)`** to change it:

```
# sysctl -w fs.file-max=65536
```

To make the change permanent, add or change the following line in the file **`/etc/sysctl.conf`**. This file is used during the boot process.

```
echo "fs.file-max=65536" >> /etc/sysctl.conf
```

Adjusting Network Settings

10.1. Changing Network Adapter Settings

To check the speed and settings of network adapters, use the **ethtool** command which works now for most network interface cards. To check the adapter settings of **eth0** run:

```
# ethtool eth0
```

To force a speed change to 1000Mbps, full duplex mode, run:

```
# ethtool -s eth0 speed 1000 duplex full autoneg off
```

To make a speed change permanent for **eth0**, set or add the **ETHTOOL_OPT** environment variable in **/etc/sysconfig/network-scripts/ifcfg-eth0**:

```
ETHTOOL_OPTS="speed 1000 duplex full autoneg off"
```

This environment variable is sourced in by the network scripts each time the network service is started.

10.2. Changing Network Kernel Settings

Oracle now uses User Datagram Protocol (UDP) as the default protocol on Linux for interprocess communication, such as cache fusion buffer transfers between the instances. However, starting with Oracle 10g network settings should be adjusted for standalone databases as well.

Oracle recommends the default and maximum send buffer size (**SO_SNDBUF** socket option) and receive buffer size (**SO_RCVBUF** socket option) to be set to 256 KB. The receive buffers are used by TCP and UDP to hold received data until it is read by the application. The receive buffer cannot overflow because the peer is not allowed to send data beyond the buffer size window. This means that datagrams will be discarded if they do not fit in the socket receive buffer. This could cause the sender to overwhelm the receiver.

The default and maximum window size can be changed in the proc file system without reboot:

The default setting in bytes of the socket receive buffer

```
# sysctl -w net.core.rmem_default=262144
```

The default setting in bytes of the socket send buffer

```
# sysctl -w net.core.wmem_default=262144
```

The maximum socket receive buffer size which may be set by using the SO_RCVBUF socket option

```
# sysctl -w net.core.rmem_max=262144
```

The maximum socket send buffer size which may be set by using the SO_SNDBUF socket option

```
# sysctl -w net.core.wmem_max=262144
```

To make the change permanent, add the following lines to the **/etc/sysctl.conf** file, which is used during the boot process:

```
net.core.rmem_default=262144
net.core.wmem_default=262144
net.core.rmem_max=262144
net.core.wmem_max=262144
```

To improve fail over performance in a RAC cluster, consider changing the following IP kernel parameters as well:

```
net.ipv4.tcp_keepalive_time
net.ipv4.tcp_keepalive_intvl
net.ipv4.tcp_retries2
net.ipv4.tcp_syn_retries
```

Changing these settings may be highly dependent on your system, network, and other applications. For suggestions, see Metalink Note:249213.1 and Note:265194.1.

On Red Hat Enterprise Linux systems the default range of IP port numbers that are allowed for TCP and UDP traffic on the server is too low for 9i and 10g systems. Oracle recommends the following port range:

```
# sysctl -w net.ipv4.ip_local_port_range="1024 65000"
```

To make the change permanent, add the following line to the /etc/sysctl.conf file, which is used during the boot process:

```
net.ipv4.ip_local_port_range=1024 65000
```

The first number is the first local port allowed for TCP and UDP traffic, and the second number is the last port number.

10.3. Flow Control for e1000 Network Interface Cards

The e1000 network interface card family do not have flow control enabled in the 2.6 kernel on Red Hat Enterprise Linux 4 and 5. If you have heavy traffic, then the RAC interconnects may lose blocks, see Metalink Bug:5058952. For more information on flow control, see Wikipedia *Flow control*[1].

To enable Receive flow control for e1000 network interface cards, add the following line to the **/etc/modprobe.conf** file:

```
options e1000 FlowControl=1
```

The e1000 module needs to be reloaded for the change to take effect. Once the module is loaded with flow control, you should see e1000 flow control module messages in **/var/log/messages**.

[1] http://en.wikipedia.org/wiki/Flow_control

Setting Shell Limits for the Oracle User

Most shells like **Bash** provide control over various resources like the maximum allowable number of open file descriptors or the maximum number of processes available to a user.

To see all shell limits, run:

```
ulimit -a
```

For more information on **ulimit** for the **Bash** shell, see man bash and search for **ulimit**.

Note

On some Linux systems setting "hard" and "soft" limits in the following examples might not work properly when you log in as user oracle via **SSH**. It might work if you log in as root and **su** to **oracle**. If you have this problem try to set **UsePrivilegeSeparation** to "no" in **/etc/ssh/sshd_config** and restart the **SSH** daemon by executing service **sshd restart**. The privilege separation does not work properly with **PAM** on some Linux systems. *Make sure to talk to the people in charge of security before disabling the **SSH** security feature "Privilege Separation".*

11.1. Limiting Maximum Number of Open File Descriptors for the Oracle User

After **/proc/sys/fs/file-max** has been changed, see *Chapter 9, Setting File Handles*, there is still a per user limit of maximum open file descriptors:

```
$ su - oracle
$ ulimit -n
1024
$
```

To change this limit, edit the **/etc/security/limits.conf** file as root and make the following changes or add the following lines, respectively:

```
oracle           soft    nofile          4096
oracle           hard    nofile          63536
```

The "soft limit" in the first line defines the number of file handles or open files that the Oracle user will have after they log in. If the Oracle user gets error messages about running out of file handles, then the Oracle user can increase the number of file handles like in this example up to 63536 ("hard limit") by executing the following command:

```
ulimit -n 63536
```

You can set the "soft" and "hard" limits higher if necessary.

Note

It is not recommend to set the "hard" limit for nofile for the oracle user equal to **/proc/sys/fs/file-max**. If you do that and the user uses up all the file handles, then the

entire system will run out of file handles. This may prevent users logging in as the system cannot open any PAM modules that are required for the login process. That is why the hard limit should be set to 63536 and not 65536.

That these limits work you also need to ensure that **pam_limits** is configured in the **/etc/pam.d/system-auth** file, or in **/etc/pam.d/sshd** for **ssh**, **/etc/pam.d/su** for **su**, or **/etc/pam.d/login** for local access and **telnet** and disable telnet for all log in methods. Here are examples of the two session entries in the **/etc/pam.d/system-auth** file:

```
session     required      /lib/security/$ISA/pam_limits.so
session     required      /lib/security/$ISA/pam_unix.so
```

Log in to the oracle user account since the changes will become effective for new login sessions only. Note the **ulimit** options are different for other shells.

```
$ su - oracle
$ ulimit -n
4096
$
```

The default limit for oracle is now 4096 and the oracle user can increase the number of file handles up to 63536:

```
$ su - oracle
$ ulimit -n
4096
$ ulimit -n 63536
$ ulimit -n
63536
$
```

To make this change permanent, you could add "**ulimit -n 63536**" ,for bash**bash**, to the **~oracle/.bash_profile** file which is the user start up file for the **bash** shell on Red Hat Enterprise Linux (to verify your shell execute **echo $SHELL**). To do this you could simply copy and paste the following commands for oracle's **bash** shell:

```
su - oracle
cat >> ~oracle/.bash_profile << EOF
ulimit -n 63536
EOF
```

To make the above changes permanent, you could also set the soft limit equal to the hard limit in **/etc/security/limits.conf**:

```
oracle           soft    nofile          63536
oracle           hard    nofile          63536
```

11.2. Limiting Maximum Number of Processes Available for the Oracle User

After reading the procedure on, *Section 11.1, "Limiting Maximum Number of Open File Descriptors for the Oracle User"* you should now have an understanding of "soft" and "hard" limits and how to change shell limits.

To see the current limit of the maximum number of processes for the oracle user, run:

```
$ su - oracle
$ ulimit -u
```

Note

The **ulimit** options are different for other shells.

To change the "soft" and "hard" limits for the maximum number of processes for the oracle user, add the following lines to the **/etc/security/limits.conf** file:

```
oracle           soft    nproc          2047
oracle           hard    nproc          16384
```

To make this change permanent, you could add "**ulimit -u 16384**",for **bash**, to the **~oracle/.bash_profile** file which is the user start up file for the bash shell on Red Hat Enterprise Linux (to verify your shell execute **echo $SHELL**). To do this you could simply copy and paste the following commands for oracle's **bash** shell:

```
su - oracle
cat >> ~oracle/.bash_profile << EOF
ulimit -u 16384
EOF
```

To make the above changes permanent, you could also set the soft limit equal to the hard limit in **/etc/security/limits.conf**:

```
oracle           soft    nproc          16384
oracle           hard    nproc          16384
```

Enabling Asynchronous I/O and Direct I/O Support

Asynchronous I/O permits Oracle to continue processing after issuing I/Os requests which leads to higher I/O performance. Red Hat Enterprise Linux also allows Oracle to issue multiple simultaneous I/O requests with a single system call. This reduces context switch overhead and allows the kernel to optimize disk activity.

To enable asynchronous I/O in Oracle Database, it is necessary to relink Oracle 9i and 10g Release 1.

Note

10g Release 2 is shipped with asynchronous I/O support enabled and does not need to be relinked. But you may have to apply a patch, see below.

12.1. Relinking Oracle9i R2 to Enable Asynchronous I/O Support

Note

Oracle 9iR2 on Red Hat Enterprise Linux 3, 4 and 5 the 9.2.0.4 patchset or higher needs to be installed together with another patch for asynchronous I/O, see Metalink Note:279069.1.

To relink Oracle9i R2 for asynchronous I/O, execute the following commands:

```
# shutdown Oracle
SQL> shutdown

su - oracle
$ cd $ORACLE_HOME/rdbms/lib
$ make -f ins_rdbms.mk async_on
$ make -f ins_rdbms.mk ioracle

# The last step creates a new "oracle" executable "$ORACLE_HOME/bin/oracle".
# It backs up the old oracle executable to $ORACLE_HOME/bin/oracleO,
# it sets the correct privileges for the new Oracle executable "oracle",
# and moves the new executable "oracle" into the $ORACLE_HOME/bin directory.
```

If asynchronous I/O needs to be disabled, execute the following commands:

```
# shutdown Oracle
SQL> shutdown

su - oracle
$ cd $ORACLE_HOME/rdbms/lib
$ make -f ins_rdbms.mk async_off
$ make -f ins_rdbms.mk ioracle
```

12.2. Relinking Oracle 10g to Enable Asynchronous I/O Support

Ensure that for 10g Release 1 and 2 the **libaio** and **libaio-devel** RPMs are installed on the system:

```
# rpm -q libaio libaio-devel
libaio-0.3.96-5
libaio-devel-0.3.96-5
```

If you relink Oracle for asynchronous I/O without installing the **libaio** RPM, then you will get an error message similar to this one:

```
SQL> connect / as sysdba
oracleorcl: error while loading shared libraries: libaio.so.1: cannot open \
shared object file: No such file or directory
ERROR:
ORA-12547: TNS:lost contact
```

The **libaio** RPMs provide a native Linux asynchronous I/O API. In other words this is a kernel accelerated asynchronous I/O for the POSIX asynchronous I/O facility.

Note

10g Release 2 is shipped with asynchronous I/O support enabled. This means that 10g Release 2 does not need to be relinked. However, there is a bug in Oracle 10.1.0.2 that causes asynchronous I/O not to be installed correctly which can result in poor DB performance, see Bug:3438751 and Note:270213.1.

To relink Oracle 10g R1 for asynchronous I/O, execute the following commands:

```
# shutdown Oracle
SQL> shutdown

su - oracle
$ cd $ORACLE_HOME/rdbms/lib
$ make PL_ORALIBS=-laio -f ins_rdbms.mk async_on
```

If asynchronous I/O needs to be disabled, run the following commands:

```
# shutdown Oracle
SQL> shutdown

su - oracle
$ cd $ORACLE_HOME/rdbms/lib
$ make -f ins_rdbms.mk async_off
```

12.3. Enabling Asynchronous I/O in Oracle 9i and 10g

If you use file systems instead of raw devices or ASM for data files, then you need to ensure that the data files reside on file systems that support asynchronous I/O (OCFS/OCFS2, ext2 and ext3). To do asynchronous I/O on file systems the **filesystemio_options** parameter needs to be set to "**asynch**".

```
filesystemio_options=asynch
```

This parameter is platform specific. By default, this parameter is set to none for Linux and thus needs to be changed:

```
SQL> show parameter filesystemio_options;

NAME                                 TYPE        VALUE
------------------------------------ ----------- -----------
filesystemio_options                 string      none
SQL>
```

The **filesystemio_options** can have the following values with Oracle9iR2:

- **asynch**: This value enables asynchronous I/O on file system files.
- **directio**: This value enables direct I/O on file system files.
- **setall**: This value enables both asynchronous and direct I/O on file system files.
- **none**: This value disables both asynchronous and direct I/O on file system files.

If you also want to enable Direct I/O Support which is available in Red Hat Enterprise Linux 4 or 5, set **filesystemio_options** to "**setall**".

In Red Hat Enterprise Linux 3

It is recommended you use direct I/O ONLY for ext2, ext3, GFS, NFS and OCFS file systems.

In Red Hat Enterprise Linux 4 and 5

It is strongly recommended to use the "**setall**" parameter for ext2, ext3, GFS, NFS and OCFS file systems.

12.4. Tuning Asynchronous I/O for Oracle 9i and 10g

For Red Hat Enterprise Linux 3 it is recommended to set **aio-max-size** to 1048576 since Oracle uses I/Os of up to 1MB. It controls the maximum I/O size for asynchronous I/Os.

Note

The **aio-max-size** tuning parameter is not applicable to the 2.6 kernel on Red Hat Enterprise Linux 4 or 5.

To determine the maximum I/O size in bytes, execute:

```
$ cat /proc/sys/fs/aio-max-size
131072
```

To change the maximum number of bytes without reboot:

```
# echo 1048576 > /proc/sys/fs/aio-max-size
```

Alternatively, you can use **sysctl(8)** to change it:

```
# sysctl -w fs.aio-max-size=1048576
```

To make the change permanent, add the following line to the **/etc/sysctl.conf** file. This file is used during the boot process:

```
$ echo "fs.aio-max-size=1048576" >> /etc/sysctl.conf
```

12.5. Verifying Asynchronous I/O Usage

To verify whether **$ORACLE_HOME/bin/oracle** was linked with asynchronous I/O, you can use the Linux commands **ldd** and **nm**.

In the following example, **$ORACLE_HOME/bin/oracle** was relinked with asynchronous I/O:

```
$ ldd $ORACLE_HOME/bin/oracle | grep libaio
 libaio.so.1 => /usr/lib/libaio.so.1 (0x0093d000)
$ nm $ORACLE_HOME/bin/oracle | grep io_getevent
 w io_getevents@@LIBAIO_0.1
$
```

In the following example, **$ORACLE_HOME/bin/oracle** has NOT been relinked with asynchronous I/O:

```
$ ldd $ORACLE_HOME/bin/oracle | grep libaio
$ nm $ORACLE_HOME/bin/oracle | grep io_getevent
 w io_getevents
$
```

If **$ORACLE_HOME/bin/oracle** is relinked with asynchronous I/O it does not necessarily mean that Oracle is really using it. You also have to ensure that Oracle is configured to use asynchronous I/O calls, see Enabling Asynchronous I/O Support.

To verify whether Oracle is making asynchronous I/O calls, you can take a look at the **/proc/slabinfo** file assuming there are no other applications performing asynchronous I/O calls on the system. This file shows kernel slab cache information in real time.

On a Red Hat Enterprise Linux 3 system where Oracle does *not* make asynchronous I/O calls, the output looks like this:

```
$ egrep "kioctx|kiocb" /proc/slabinfo
kioctx                 0      0    128    0    0    1 : 1008  252
kiocb                  0      0    128    0    0    1 : 1008  252
$
```

Once Oracle makes asynchronous I/O calls, the output on a Red Hat Enterprise Linux 3 system will look like this:

```
$ egrep "kioctx|kiocb" /proc/slabinfo
kioctx               690    690    128   23   23    1 : 1008  252
kiocb              58446  65160    128 1971 2172    1 : 1008  252
```

```
$
```

The numbers in red (number of active objects) show whether Oracle makes asynchronous I/O calls. The output will look a little bit different in Red Hat Enterprise Linux 4 and 5. However, the numbers in red will show same behavior in Red Hat Enterprise Linux 3 and Red Hat Enterprise Linux 4 and 5. The first column displays the cache names kioctx and kiocb. The second column shows the number of active objects currently in use. And the third column shows how many objects are available in total, used and unused.

To see kernel slab cache information in real time, you can also use the slabtop command:

```
$ slabtop
Active / Total Objects (% used)    : 293568 / 567030 (51.8%)
Active / Total Slabs (% used)      : 36283 / 36283 (100.0%)
Active / Total Caches (% used)     : 88 / 125 (70.4%)
Active / Total Size (% used)       : 81285.56K / 132176.36K (61.5%)
Minimum / Average / Maximum Object : 0.01K / 0.23K / 128.00K

OBJS    ACTIVE  USE    OBJ SIZE  SLABS OBJ/SLAB CACHE   SIZE NAME
178684  78396   43%    0.12K     5764    31     23056K  size-128
127632  36292   28%    0.16K     5318    24     21272K  dentry_cache
102815  74009   71%    0.69K     20563   5      82252K  ext3_inode_cache
71775   32434   45%    0.05K     957     75     3828K   buffer_head
19460   15050   77%    0.27K     1390    14     5560K   radix_tree_node
13090   13015   99%    0.03K     110     119    440K    avtab_node
12495   11956   95%    0.03K     105     119    420K    size-32
...
```

Slab caches are a special memory pool in the kernel for adding and removing objects, such as data structures or data buffers, of the same size. Its a cache for commonly used objects where the kernel does not have to re-allocate and initialize the object each time it is being reused, and free the object each time it is being destroyed. The slab allocator scheme basically prevents memory fragmentation and it prevents the kernel from spending too much time allocating, initializing, and freeing the same objects.

Configuring I/O for Raw Partitions

13.1. General

Raw partitions allow Oracle to bypass the OS cache. A raw device can be assigned or bound to block devices such as disk or disk partitions. When a raw device is bound to a disk, any read or write access to the raw device will perform a raw I/O with the disk. A raw I/O through the **`/dev/raw`** interface bypasses the kernel's block buffer cache entirely that is normally associated with block devices and goes right to the low level device itself. By bypassing the cache it accesses a physical device directly which allows applications such as Oracle databases to have more control over the I/O to the physical device. In fact, Oracle does it is own data caching and raw devices allow Oracle to ensure that data gets written to the disk immediately without OS caching.

Since Automatic Storage Management (ASM) is the recommended option for large amounts of storage in RAC environments, the focus of this article and section is on the usage of raw devices and block devices for ASM. ASM offers many advantages over conventional file systems. The ASM file system is not buffered and supports asynchronous I/O. It allows you to group sets of physical disks to logical entities as disk groups. You can add or remove disks without downtime. In fact, you could move a whole database from one SAN storage to another SAN without downtime. ASM spreads I/O over all the available disks automatically to avoid hot spots. ASM does also it is own striping and offers mirroring. ASM can be setup using the ASM library driver or raw devices. Starting with 10g R2, neither is necessarily required.

Note

Since raw I/O is now being deprecated by the Linux community and Red Hat Enterprise Linux 4 and 5, Oracle 10g R2 no longer requires raw devices for the database. Oracle 10g R2 automatically opens all block devices such as SCSI disks using the **`O_DIRECT`** flag, thus bypasses the OS cache. But for older Oracle Database and Red Hat Enterprise Linux versions raw devices are still a recommended option for ASM and data files. For more information on using block devices, see *Section 13.4, "Using Block Devices for Oracle 10g Release 2 in Red Hat Enterprise Linux 4 and 5"*. Unfortunately, Oracle **Clusterware R2 OUI** still requires raw devices or a Cluster File System.

Caution

The name of the devices are assigned by the Linux and is determined by the scan order of the bus. Therefore, the device names are not guaranteed to persist across reboots. For example, SCSI device **`/dev/sdb`** can change to **`/dev/sda`** if the scan order of the controllers is not configured. To force the scan order of the controllers, aliases can be set in **`/etc/modprobe.conf`**. For example:

```
alias scsi_hostadapter1 aic7xxx

alias scsi_hostadapter2 lpfc
```

These settings will guarantee that the Adaptec adapter for local storage is used first and then the Emulex adapter(s) for SAN storage. Fortunately, Red Hat Enterprise Linux 4 and 5 have already addressed this issue by delaying the loading of lpfc (Emulex) and various qla (QLogic) drivers until after all other SCSI devices have been loaded. This means that

the alias settings in this example would not be required in Red Hat Enterprise Linux 4 and 5. For more information, see *Red Hat Enterprise Linux AS 4 Release Notes*[1]. Be also careful when adding/removing devices which can change device names on the system. Starting Oracle with incorrect device names or raw devices can cause permanent damage to the database. For stable device naming in Linux 2.4 and 2.6, see *Optimizing Linux I/O*[2].

13.2. Basics of Raw Devices

To bind the first raw device **/dev/raw/raw1** to the **/dev/sdz** SCSI disk or LUN you can execute the following command:

```
# raw /dev/raw/raw1 /dev/sdz
```

Now when you run the **dd** command on **/dev/raw/raw1**, it will write directly to **/dev/sdz** bypassing the OS block buffer cache:

Warning

The following command will overwrite data on **dev/sdz**

```
# dd if=/dev/zero of=/dev/sdz count=1
```

To permanently bind **/dev/raw/raw1** to **/dev/sdz**, add an entry to the **/etc/sysconfig/rawdevices** file:

```
/dev/raw/raw1 /dev/sdz
```

Now when you run **/etc/init.d/rawdevices** it will read the **/etc/sysconfig/rawdevices** file and execute the **raw** command for each entry:

```
/etc/init.d/rawdevices start
```

To have **/etc/init.d/rawdevices** run each time the system boot, it can be activated by executing the following command:

```
chkconfig rawdevices on
```

Note

For each block device you need to use another raw device.

To bind the third raw device to the second partition of **/dev/sdz**, the entry in **/etc/sysconfig/rawdevices** would look like this:

```
/dev/raw/raw3 /dev/sdz2
```

Or to bind the 100th raw device to **/dev/sdz**, the entry in **/etc/sysconfig/rawdevices** would look like this:

```
/dev/raw/raw100 /dev/sdz
```

13.3. Using Raw Devices for Oracle Databases

Many guides and documentations show instructions on using the devices in **/dev/raw/** for configuring raw devices for data files. It is not recommend to use the raw devices in **/dev/raw/** for the following reason: When you configure raw devices for Oracle data files, you also have to change ownership and permissions of the devices in **/dev/raw/** to allow Oracle to read and write to these raw devices. But all device names in **/dev/raw/** are owned by the dev RPM. So when the Linux systems administrator upgrades the **dev** RPM, which may happen as part of an operating system update, then all device names in **/dev/raw/** will automatically be recreated. This means that ownership and permissions must be set each time the dev RPM gets upgraded. Therefore it is recommend to create all raw devices for Oracle datafiles in an Oracle data directory such as **/u02**.

For example, to create a new raw device for the system data file **system01.dbf** in **/u02/orcl/**, execute the following command:

```
# mknod /u02/orcl/system01.dbf c 162 1
```

This command creates a new raw device called **/u02/orcl/system01.dbf** with minor number 1, which is equivalent to the first raw device **/dev/raw/raw1**. The major number 162 designates the device as a raw device. A major number always identifies the driver associated with the device.

To grant **oracle:dba** read and write permissions, execute:

```
# chown oracle.dba /u02/orcl/system01.dbf
# chown 660 /u02/orcl/system01.dbf
```

To bind this new raw device to the first partition of **/dev/sdb**, add the following line to the **/etc/sysconfig/rawdevices** file:

```
/u02/orcl/system01.dbf /dev/sdb1
```

To activate the raw device, execute:

```
/etc/init.d/rawdevices start
```

Here is an example for creating raw devices for ASM:

```
# mknod /u02/oradata/asmdisks/disk01 c 162 1
# mknod /u02/oradata/asmdisks/disk02 c 162 2
# mknod /u02/oradata/asmdisks/disk03 c 162 3
# mknod /u02/oradata/asmdisks/disk03 c 162 4

# chown oracle.dba /u02/oradata/asmdisks/disk01
# chown oracle.dba /u02/oradata/asmdisks/disk02
# chown oracle.dba /u02/oradata/asmdisks/disk03
# chown oracle.dba /u02/oradata/asmdisks/disk04

# chmod 660 /u02/oradata/asmdisks/disk01
# chmod 660 /u02/oradata/asmdisks/disk02
# chmod 660 /u02/oradata/asmdisks/disk03
# chmod 660 /u02/oradata/asmdisks/disk04
```

And the **/etc/sysconfig/rawdevices** file would look something like this if you use EMC PowerPath:

```
/u02/oradata/asmdisks/disk01 /dev/emcpowera
/u02/oradata/asmdisks/disk02 /dev/emcpowerb
/u02/oradata/asmdisks/disk03 /dev/emcpowerc
/u02/oradata/asmdisks/disk04 /dev/emcpowerd
```

In this example, 4 raw devices have been created using minor numbers 1 through 4. This means that the devices **/dev/raw/raw1../dev/raw/raw4** should not be used by any application on the system. But this should not be an issue since all raw devices should be configured in one place, which is the **/etc/sysconfig/rawdevices** file. Note that you could also partition the LUNs or disks and configure a raw device for each disk partition.

13.4. Using Block Devices for Oracle 10g Release 2 in Red Hat Enterprise Linux 4 and 5

For Oracle 10g Release 2 in Red Hat Enterprise Linux 4 and 5 it is not recommended to use raw devices but to use block devices instead. Raw I/O is still available in Red Hat Enterprise Linux 4 and 5, but it is now a deprecated interface. In fact, raw I/O has been deprecated by the Linux community. It has been replaced by the **O_DIRECT** flag, which can be used for opening block devices to bypass the operating system's cache. Unfortunately, **Oracle Clusterware R2 OUI** has not been updated and still requires raw devices or a Cluster File System. There is also another bug, see bug number *5021707*[3].

By default, reading and writing to block devices are buffered I/Os. **Oracle Database 10g R2** now automatically opens all block devices such as SCSI disks using the **O_DIRECT flag**, thus bypassing the OS cache. For example, when you create disk groups for ASM and you want to use the SCSI block devices **/dev/sdb** and **/dev/sdc**, you can simply set the Disk Discovery Path to "**/dev/sdb, /dev/sdc**" to create the ASM disk group. There is no need to create raw devices and to point the Disk Discovery Path to it.

Using the ASM example from *Section 13.3, "Using Raw Devices for Oracle Databases"*, the Oracle data directory could be setup the following way:

```
$ ln -s /dev/emcpowera /u02/oradata/asmdisks/disk01
$ ln -s /dev/emcpowerb /u02/oradata/asmdisks/disk02
$ ln -s /dev/emcpowerc /u02/oradata/asmdisks/disk03
$ ln -s /dev/emcpowerd /u02/oradata/asmdisks/disk04
```

And the following command needs to be executed after each reboot:

```
# chown oracle.dba /u02/oradata/asmdisks/*
```

You need to ensure that the ownership of block devices is changed to **oracle:dba** or **oracle:oinstall**. Otherwise Oracle can not access the block devices and ASM disk discovery will not list them. You also need to ensure that the ownership of block devices is set after each reboot since Linux changes the ownership of block devices back to "**brw-rw---- 1 root disk**" at boot time.

[3] http://www.oracle.com/technology/tech/linux/validated-configurations/html/vc_dell6850-rhel4-cx500-1_1.html

Large Memory Optimization, Big Pages, and Huge Pages

Please Note

As explained in detail in this section, enabling big pages helps reduce TLB misses. However, this performance benefit is realized primarily when using large SGA sizes. Once a portion of memory is locked down for big pages, applications that use normal pages cannot access that portion of the memory. It is very important to make sure that there is enough memory for normal pages for applications and users to avoid excessive swapping. So, it is recommended that big pages be used only on systems that have large amounts of physical memory and for SGA sizes of 16GB or greater.

Big Pages in Red Hat Enterprise Linux 2.1 and Huge Pages in Red Hat Enterprise Linux 3, 4 and 5 are very useful for large Oracle SGA sizes and in general for systems with large amount of physical memory. It optimizes the use of Translation Lookaside Buffers (TLB), locks these larger pages in RAM, and the system has less bookkeeping work to do for that part of virtual memory due to larger page sizes. This is a useful feature that should be used on x86 and x86-64 platforms. The default page size in Linux for x86 is 4KB.

Physical memory is partitioned into pages which are the basic unit of memory management. When a Linux process accesses a virtual address, the CPU must translate it into a physical address. Therefore, for each Linux process the kernel maintains a page table which is used by the CPU to translate virtual addresses into physical addresses. But before the CPU can do the translation it has to perform several physical memory reads to retrieve page table information. To speed up this translation process for future references to the same virtual address, the CPU saves information for recently accessed virtual addresses in its Translation Lookaside Buffers (TLB) which is a small but very fast cache in the CPU. The use of this cache makes virtual memory access very fast. Since TLB misses are expensive, TLB hits can be improved by mapping large contiguous physical memory regions by a small number of pages. So fewer TLB entries are required to cover larger virtual address ranges. A reduced page table size also means a reduction in memory management overhead. To use larger page sizes for shared memory, Big Pages (Red Hat Enterprise Linux 2.1) or Huge Pages (Red Hat Enterprise Linux 3, 4 and 5) must be enabled which also locks these pages in physical memory.

14.1. Big Pages in Red Hat Enterprise Linux 2.1 and Huge Pages in Red Hat Enterprise Linux 3

In Red Hat Enterprise Linux 2.1 large memory pages can be configured using the Big Pages, **`bigpages`**, feature. In Red Hat Enterprise Linux 3 or 4 Red Hat replaced Big Pages with a feature called Huge Pages, **`hugetlb`**, which behaves a little bit different. The Huge Pages feature in Red Hat Enterprise Linux 3 or 4 allows you to dynamically allocate large memory pages without a reboot. Allocating and changing Big Pages in Red Hat Enterprise Linux 2.1 always required a reboot. However, if memory gets too fragmented in Red Hat Enterprise Linux 3 or 4 allocation of physically contiguous memory pages can fail and a reboot may become necessary.

The advantages of Big Pages and Huge Pages for database performance are:

- increased performance through increased Translation Lookaside Buffer (TLB) hits.

- pages are locked in memory and are never swapped out which guarantees that shared memory such as SGA remains in RAM.
- contiguous pages are pre-allocated and cannot be used for anything else but for System V shared memory, for example SGA.
- less bookkeeping work for the kernel in that part of virtual memory due to larger page sizes.

14.2. Usage of Big Pages and Huge Pages in Oracle 9i and 10g

Big pages are supported implicitly in Red Hat Enterprise Linux 2.1. But Huge Pages in Red Hat Enterprise Linux 3, 4 and 5 need to be requested explicitly by the application by using the **SHM_HUGETLB** flag when invoking the **shmget()** system call. This ensures that shared memory segments are allocated out of the Huge Pages pool. This is done automatically in Oracle 10g and 9i R2 (9.2.0.6) but earlier Oracle 9i R2 versions require a patch, see Metalink Note:262004.1.

14.3. Sizing Big Pages and Huge Pages

With the Big Pages and Huge Pages feature you specify how many physically contiguous large memory pages should be allocated and pinned in RAM for shared memory like Oracle SGA. For example, if you have three Oracle instances running on a single system with 2 GB SGA each, then at least 6 GB of large pages should be allocated. This will ensure that all three SGAs use large pages and remain in main physical memory. Furthermore, if you use ASM on the same system, then it is prudent to add an additional 200MB. I have seen ASM instances creating between 70 MB and 150 MB shared memory segments. And there might be other non-Oracle processes that allocate shared memory segments as well.

It is, however, not recommended to allocate too many Big or Huge Pages. These pre-allocated pages can only be used for shared memory. This means that unused Big or Huge Pages will not be available for other use than for shared memory allocations even if the system runs out of memory and starts swapping. Also take note that Huge Pages are not used for the **ramfs** shared memory file system, see *Section 14.8, “Huge Pages and Shared Memory File System in Red Hat Enterprise Linux 3”*, but Big Pages can be used for the **shm** file system in Red Hat Enterprise Linux 2.1.

14.4. Checking Shared Memory Before Starting Oracle Databases

It is very important to always check the shared memory segments before starting an instance. An abandoned shared memory segment, from an instance crash for example, is not removed, it will remain allocated in the Big Pages or Huge Pages pool. This could mean that new allocated shared memory segments for the new instance SGA will not fit into the Big Pages or Huge Pages pool. For more information on removing shared memory, see *Section 7.4, “Removing Shared Memory”*.

14.5. Configuring Big Pages in Red Hat Enterprise Linux 2.1

Before configuring Big Pages, ensure to have read *Section 14.3, “Sizing Big Pages and Huge Pages”*.

Note that Big Pages in x86 Red Hat Enterprise Linux 2.1 can only be allocated and pinned above approximately 860MB of physical RAM which is known as **Highmem** or high memory region in x86.

Thus, Big Pages cannot be larger than **Highmem**. The total amount of memory in the high region can be obtained by reading the memory statistic **HighTotal** from the **/proc/meminfo** file:

```
$ grep "HighTotal" /proc/meminfo
HighTotal:     9043840 kB
$
```

The Big Pages feature can be enabled with the following command:

```
# echo "1" > /proc/sys/kernel/shm-use-bigpages
```

Alternatively, you can use **sysctl(8)** to change it:

```
# sysctl -w kernel.shm-use-bigpages=1
```

To make the change permanent, add the following line to the file **/etc/sysctl.conf**. This file is used during the boot process.

```
echo "kernel.shm-use-bigpages=1" >> /etc/sysctl.conf
```

Setting **kernel.shm-use-bigpages** to 2 enables the Big Pages feature for the shared memory file system (**shmfs**). Setting **kernel.shm-use-bigpages** to 0 disables the Big Pages feature. In Red Hat Enterprise Linux 2.1 the size of the Big Pages pool is configured by adding a parameter to the kernel boot command. For example, if you use **GRUB** and you want to set the Big Pages pool to 1000 MB, edit the **/etc/grub.conf** file and add the "**bigpages**" parameter as follows:

```
default=0
timeout=10
title Red Hat Linux Advanced Server (2.4.9-e.40enterprise)
 root (hd0,0)
 kernel /vmlinuz-2.4.9-e.40enterprise ro root=/dev/sda2 bigpages=1000MB
 initrd /initrd-2.4.9-e.40enterprise.img
title Red Hat Linux Advanced Server (2.4.9-e.40smp)
 root (hd0,0)
 kernel /vmlinuz-2.4.9-e.40smp ro root=/dev/sda2
 initrd /initrd-2.4.9-e.40smp.img
```

After this change the system must be rebooted:

```
# shutdown -r now
```

After a system reboot the 1000 MB Big Pages pool should show up under **BigPagesFree** in **/proc/meminfo**.

```
grep BigPagesFree /proc/meminfo
```

Note that if **HighTotal** in **/proc/meminfo** is 0 KB, then **BigPagesFree** will always be 0 KB as well since Big Pages can only be allocated and pinned above approximately 860MB of physical RAM.

14.6. Configuring Huge Pages in Red Hat Enterprise Linux 3

Before configuring Big Pages, ensure to have read *Section 14.3, "Sizing Big Pages and Huge Pages"*.

In Red Hat Enterprise Linux 3 the desired size of the Huge Pages pool is specified in megabytes. The size of the pool should be configured by the incremental size of the Huge Page size. To obtain the size of Huge Pages, execute the following command:

```
$ grep Hugepagesize /proc/meminfo
Hugepagesize:     2048 kB
$
```

The number of Huge Pages can be configured and activated by setting **hugetlb_pool** in the **proc** file system. For example, to allocate a 1GB Huge Page pool, execute:

```
# echo 1024 > /proc/sys/vm/hugetlb_pool
```

Alternatively, you can use **sysctl(8)** to change it:

```
# sysctl -w vm.hugetlb_pool=1024
```

To make the change permanent, add the following line to the file **/etc/sysctl.conf**. This file is used during the boot process. The Huge Pages pool is usually guaranteed if requested at boot time:

```
# echo "vm.hugetlb_pool=1024" >> /etc/sysctl.conf
```

If you allocate a large number of Huge Pages, the execution of the above commands can take a while. To verify whether the kernel was able to allocate the requested number of Huge Pages, execute:

```
$ grep HugePages_Total /proc/meminfo
HugePages_Total:   512
$
```

The output shows that 512 Huge Pages have been allocated. Since the size of Huge Pages on this system is 2048 KB, a Huge Page pool of 1GB has been allocated and pinned in physical memory.

If **HugePages_Total** is lower than what was requested with **hugetlb_pool**, then the system does either not have enough memory or there are not enough physically contiguous free pages. In the latter case the system needs to be rebooted which should give you a better chance of getting the memory.

To get the number of free Huge Pages on the system, execute:

```
$ grep HugePages_Free /proc/meminfo
```

Free system memory will automatically be decreased by the size of the Huge Pages pool allocation regardless whether the pool is being used by an application like Oracle database or not being used:

```
$ grep MemFree /proc/meminfo
```

After an Oracle database starts up, verify the Huge Pages usage. The number of free Huge Pages should decrease.

```
$ grep HugePages_Free /proc/meminfo
```

To free the Huge Pages pool, you can execute:

```
# echo 0 > /proc/sys/vm/hugetlb_pool
```

This command usually takes a while to finish.

14.7. Configuring Huge Pages in Red Hat Enterprise Linux 4 or 5

Before configuring Big Pages, ensure to have read *Section 14.3, "Sizing Big Pages and Huge Pages"*.

In Red Hat Enterprise Linux 4 or 5 the size of the Huge Pages pool is specified by the desired number of Huge Pages. To calculate the number of Huge Pages you first need to know the Huge Page size. To obtain the size of Huge Pages, execute the following command:

```
$ grep Hugepagesize /proc/meminfo
Hugepagesize:     2048 kB
$
```

The output shows that the size of a Huge Page on this system is 2MB. This means if a 1GB Huge Pages pool should be allocated, then 512 Huge Pages need to be allocated. The number of Huge Pages can be configured and activated by setting **nr_hugepages** in the proc file system. For example, to allocate 512 Huge Pages, execute:

```
# echo 512 > /proc/sys/vm/nr_hugepages
```

Alternatively, you can use **sysctl(8)** to change it:

```
# sysctl -w vm.nr_hugepages=512
```

To make the change permanent, add the following line to the file **/etc/sysctl.conf**. This file is used during the boot process. The Huge Pages pool is usually guaranteed if requested at boot time:

```
# echo "vm.nr_hugepages=512" >> /etc/sysctl.conf
```

If you allocate a large number of Huge Pages, the execution of the above commands can take a while. To verify whether the kernel was able to allocate the requested number of Huge Pages, run:

```
$ grep HugePages_Total /proc/meminfo
HugePages_Total:   512
$
```

The output shows that 512 Huge Pages have been allocated. Since the size of Huge Pages is 2048 KB, a Huge Page pool of 1GB has been allocated and pinned in physical memory.

If **HugePages_Total** is lower than what was requested with **nr_hugepages**, then the system does either not have enough memory or there are not enough physically contiguous free pages. In the latter case the system needs to be rebooted which should give you a better chance of getting the memory.

To get the number of free Huge Pages on the system, execute:

```
$ grep HugePages_Free /proc/meminfo
```

Free system memory will automatically be decreased by the size of the Huge Pages pool allocation regardless whether the pool is being used by an application like Oracle DB or not:

```
$ grep MemFree /proc/meminfo
```

Note

In order that an Oracle database can use Huge Pages in Red Hat Enterprise Linux 4 or 5, you also need to increase the **ulimit** parameter "**memlock**" for the oracle user in **/etc/security/limits.conf** if "**max locked memory**" is not unlimited or too small, see ulimit -a or ulimit -l. An example can be seen below.

```
oracle           soft    memlock         1048576
oracle           hard    memlock         1048576
```

The **memlock** parameter specifies how much memory the oracle user can lock into its address space. Note that Huge Pages are locked in physical memory. The memlock setting is specified in KB and must match the memory size of the number of Huge Pages that Oracle should be able to allocate. So if the Oracle database should be able to use 512 Huge Pages, then memlock must be set to at least 512 * **Hugepagesize**, which on this system would be 1048576 KB (512*1024*2). If memlock is too small, then no single Huge Page will be allocated when the Oracle database starts. For more information on setting shell limits, see *Chapter 22, Setting Shell Limits for Your Oracle User*.

Log in as the oracle user again and verify the new memlock setting by executing **ulimit -l** before starting the database.

After an Oracle DB startup you can verify the usage of Huge Pages by checking whether the number of free Huge Pages has decreased:

```
$ grep HugePages_Free /proc/meminfo
```

To free the Huge Pages pool, you can execute:

```
# echo 0 > /proc/sys/vm/nr_hugepages
```

This command usually takes a while to finish.

14.8. Huge Pages and Shared Memory File System in Red Hat Enterprise Linux 3

The following example shows that the Huge Pages pool is not being used by the **ramfs** shared memory file systems. The **ramfs** shared memory file systems can be used for Configuring Very Large Memory (VLM).

The **ipcs** command shows only System V shared memory segments. It does not display shared memory of a shared memory file systems. The following command shows System V shared memory segments on a node running a database with an SGA of 2.6 GB:

```
# ipcs -m

------ Shared Memory Segments --------
key        shmid      owner      perms          bytes      nattch     status
0x98ab8248 1081344    oracle       600       77594624           0
0xe2e331e4 1245185    oracle       600     2736783360           0
```

The first shared memory segment of 74 MB was created by the ASM instance. The second shared memory segment of 2.6 GB was created by the database instance.

On this database system the size of the database buffer cache is 2 GB:

```
db_block_buffers = 262144
db_block_size    = 8192
```

The following command shows that Oracle allocated a shared memory file of 2GB (262144*8192=2147483648) for the buffer cache on the **ramfs** shared memory file system:

```
# mount | grep ramfs
ramfs on /dev/shm type ramfs (rw)
# ls -al /dev/shm
total 204
drwxr-xr-x    1 oracle   dba               0   Oct 30 16:00 .
drwxr-xr-x   22 root     root         204800   Oct 30 16:00 ..
-rw-r-----    1 oracle   dba      2147483648   Nov  1 16:46 ora_orcl1_1277954
```

The next command shows how many Huge Pages are currently being used on this system:

```
$ grep Huge /proc/meminfo
HugePages_Total:  1536
HugePages_Free:    194
Hugepagesize:     2048 kB
$
```

The output shows that 1342 (1536-194) Huge Pages are being used. This translates into 2814377984 (1342*2048*1024) bytes being allocated in the Huge Pages pool. This number matches the size of both shared memory segments (2736783360+77594624=2814377984) displayed by the **ipcs** command above.

This shows that the Huge Pages pool is not being used for the **ramfs** shared memory file system. Hence, you do not need to increase the Huge Pages pool if you use the **ramfs** shared memory file system.

Growing the Oracle SGA to 2.7 GB in x86 Red Hat Enterprise Linux 2.1 Without VLM

15.1. General

Due to 32 bit virtual address limitations workarounds have been implemented in Linux to increase the maximum size for shared memories. The workaround is to lower the Mapped Base Address (**mapped_base**) for shared libraries and the SGA Attach Address for shared memory segments. Lowering the Mapped Base Address and the SGA Attach Address allows SGA sizes up to 2.7 GB. By default, the shared memory segment size can only be increased to roughly 1.7 GB in Red Hat Enterprise Linux 2.1.

To better understand the process of lowering the Mapped Base Address for shared libraries and the SGA Attach Address for shared memory segments, a basic understanding of the Linux memory layout is necessary.

15.2. Linux Memory Layout

The 4 GB address space in 32 bit x86 Linux is usually split into different sections for every process on the system:

- 0GB-1GB User space - Used for text, code and **brk**/**sbrk** allocations. **malloc** uses **brk** for small chunks.
- 1GB-3GB User space - Used for shared libraries, shared memory, and the stack. Shared memory and **malloc** use **mmap**. **malloc** uses **mmap** for large chunks.
- 3GB-4GB Kernel Space - Used by and for the kernel itself

In older Linux systems the split between **brk(2)** and **mmap(2)** was changed by setting the kernel parameter **TASK_UNMAPPED_BASE** and by recompiling the kernel. However, on all Red Hat Enterprise Linux systems this parameter can be changed dynamically as will be shown later. The **mmap** allocated memory grow bottom up from 1GB and the stack grows top down from around 3GB. The split between userspace and kernelspace is set by the kernel parameter **PAGE_OFFSET** which is usually **0xc0000000** (3GB).

By default, in Red Hat Enterprise Linux 2.1 the address space between **0x40000000** (1 GB) and **0xc0000000** (3 GB) is available for mapping shared libraries and shared memory segments. The default mapped base for loading shared libraries is **0x40000000** (1 GB) and the SGA attach address for shared memory segments is above the shared libraries. In Oracle 9i on Red Hat Enterprise Linux 2.1 the default SGA attach address for shared memory is **0x50000000** (1.25 GB) where the SGA is mapped. This leaves 0.25 GB space for loading shared libraries between **0x40000000** (1 GB) and **0x50000000** (1.25 GB).

The address mappings of processes can be checked by viewing the proc file **/proc/<pid>/maps** where pid stands for the process ID. Here is an example of a default address mapping of an Oracle 9i process in Red Hat Enterprise Linux 2.1:

```
08048000-0ab11000 r-xp 00000000 08:09 273078 /ora/product/9.2.0/bin/oracle
```

```
0ab11000-0ab99000 rw-p 02ac8000 08:09 273078 /ora/product/9.2.0/bin/oracle
0ab99000-0ad39000 rwxp 00000000 00:00 0
40000000-40016000 r-xp 00000000 08:01 16     /lib/ld-2.2.4.so
40016000-40017000 rw-p 00015000 08:01 16     /lib/ld-2.2.4.so
40017000-40018000 rw-p 00000000 00:00 0
40018000-40019000 r-xp 00000000 08:09 17935  /ora/product/9.2.0/lib/libodmd9.so
40019000-4001a000 rw-p 00000000 08:09 17935  /ora/product/9.2.0/lib/libodmd9.so
4001a000-4001c000 r-xp 00000000 08:09 16066  /ora/product/9.2.0/lib/libskgxp9.so
...
42606000-42607000 rw-p 00009000 08:01 50     /lib/libnss_files-2.2.4.so
50000000-50400000 rw-s 00000000 00:04 163842 /SYSV00000000 (deleted)
51000000-53000000 rw-s 00000000 00:04 196611 /SYSV00000000 (deleted)
53000000-55000000 rw-s 00000000 00:04 229380 /SYSV00000000 (deleted)
...
bfffb000-c0000000 rwxp ffffc000 00:00 0
```

As this address mapping shows, shared libraries start at **0x40000000** (1 GB) and System V shared memory, in this case SGA, starts at **0x50000000** (1.25 GB). Here is a summary of all the entries:

The text (code) section is mapped at 0x08048000:

```
08048000-0ab11000 r-xp 00000000 08:09 273078  /ora/product/9.2.0/bin/oracle
```

The data section is mapped at 0x0ab11000:

```
0ab11000-0ab99000 rw-p 02ac8000 08:09 273078  /ora/product/9.2.0/bin/oracle
```

The uninitialized data segment .bss is allocated at 0x0ab99000:

```
0ab99000-0ad39000 rwxp 00000000 00:00 0
```

The base address for shared libraries is 0x40000000:

```
40000000-40016000 r-xp 00000000 08:01 16      /lib/ld-2.2.4.so
```

The base address for System V shared memory, in this case SGA, is 0x50000000:

```
50000000-50400000 rw-s 00000000 00:04 163842  /SYSV00000000 (deleted)
```

The stack is allocated at 0xbfffb000:

```
bfffb000-c0000000 rwxp ffffc000 00:00 0
```

15.3. Increasing Space for the SGA in Red Hat Enterprise Linux 2.1

To increase the maximum default size of shared memory for the SGA from 1.7 GB to 2.7GB, the Mapped Base Address (**mapped_base**) for shared libraries must be lowered from **0x40000000** (1 GB) to **0x10000000** (0.25 GB) and the SGA Attach Address for shared memory segments must be lowered from **0x50000000** (1.25 GB) to **0x15000000** (336 MB). Lowering the SGA attach address increases the available space for shared memory almost 1 GB. If shared memory starts at **0x15000000** (336 MB), then the space between **0x15000000** (336 MB) and **0xc0000000** (3GB) minus stack size becomes available for the SGA. Note the mapped base for shared libraries should

not be above the SGA attach address, example, between **0x15000000** (336 MB) and **0xc0000000** (3GB).

To increase the space for shared memory in Red Hat Enterprise Linux 2.1, the mapped base for shared libraries for the Oracle processes must be changed by root. And the oracle user must relink Oracle to relocate or lower the SGA attach address for shared memory segments.

15.4. Lowering the Mapped Base Address for Shared Libraries in Red Hat Enterprise Linux 2.1

The default mapped base address for shared libraries in Red Hat Enterprise Linux 2.1 is **0x40000000** (1 GB). To lower the mapped base for a Linux process, the file **/proc/<pid>/mapped_base** must be changed where **<pid>** stands for the process ID. This means that his is not a system wide parameter. In order to change the mapped base for Oracle processes, the address mapping of the parent shell terminal session that spawns Oracle processes (instance) must be changed for the child processes to inherit the new mapping.

Login as oracle and run the following command to obtain the process ID of the shell where **sqlplus** will later be executed:

```
$ echo $$
```

Login as root in another shell terminal session and change the mapped_base for this process ID to **0x10000000** (decimal 268435456):

```
# echo 268435456 > /proc/<pid>/mapped_base
```

Now when Oracle processes are started with **sqlplus** in this shell, they will inherit the new mapping. But before Oracle can be started, the SGA Attach Address for shared memory must be lowered as well.

15.5. Lowering the SGA Attach Address for Shared Memory Segments in Oracle 9i

The default SGA attach address for shared memory segments in Oracle 9i on Red Hat Enterprise Linux 2.1 is 0x50000000 (1.25 GB). To lower the SGA attach address for shared memory, the Oracle utility genksms must be used before the relinking:

Note

The examples below use # to represent comments not a root shell.

Login as oracle and execute the following commands:

```
# shutdown Oracle
SQL> shutdown

cd $ORACLE_HOME/rdbms/lib

# Make a backup of the ksms.s file if it exists
```

```
[[ ! -f ksms.s_orig ]] && cp ksms.s ksms.s_orig

# Modify the SGA attach address in the ksms.s file before relinking Oracle
genksms -s 0x15000000 > ksms.s
```

Rebuild the Oracle executable by entering the following commands:

```
# Create a new ksms object file
make -f ins_rdbms.mk ksms.o

# Create a new "oracle" executable ($ORACLE_HOME/bin/oracle):
make -f ins_rdbms.mk ioracle

# The last step creates a new Oracle binary in $ORACLE_HOME/bin
# that loads the SGA at the address specified by sgabeg in ksms.s:
#   .set   sgabeg,0X15000000
```

Now when Oracle is started in the shell terminal session for which the mapped_base for shared libraries was changed at *Section 15.4, "Lowering the Mapped Base Address for Shared Libraries in Red Hat Enterprise Linux 2.1"*, the SGA attach address for Oracle's shared memory segments and hence SGA can be displayed with the following commands:

```
# Get pid of e.g. the Oracle checkpoint process
$ /sbin/pidof ora_dbw0_$ORACLE_SID
13519
$ grep '.so' /proc/13519/maps |head -1
10000000-10016000 r-xp 00000000 03:02 750738     /lib/ld-2.2.4.so
$ grep 'SYS' /proc/13519/maps |head -1
15000000-24000000 rw-s 00000000 00:04 262150     /SYSV3ecee0b0 (deleted)
$
```

The SGA size can now be increased to approximately 2.7 GB. If you create the SGA larger than 2.65 GB, then test the database very thoroughly to ensure no memory allocation problems arise.

15.6. Allowing the Oracle User to Change the Mapped Base Address for Shared Libraries

As shown at *Section 15.4, "Lowering the Mapped Base Address for Shared Libraries in Red Hat Enterprise Linux 2.1"* only root can change the mapped_base for shared libraries. Using sudo we can give the "oracle" user the privilege to change the mapped base for shared libraries for the shell terminal session without providing full root access to the system.

The procedure is as follows:

Create a script called "**`/usr/local/bin/ChangeMappedBase`**" which changes the mapped_base for shared libraries for for its own shell:

```
# cat /usr/local/bin/ChangeMappedBase
#/bin/sh
echo 268435456 > /proc/$PPID/mapped_base
```

Make the script executable:

```
# chown root.root /usr/local/bin/ChangeMappedBase
# chmod 755 /usr/local/bin/ChangeMappedBase
```

Allow the oracle user to execute **/usr/local/bin/ChangeMappedBase** via **sudo** without password:

```
# echo "oracle   ALL=NOPASSWD: /usr/local/bin/ChangeMappedBase" >> \
 /etc/sudoers
```

Now the Oracle user can run **/usr/local/bin/ChangeMappedBase** to change the mapped_base for its own shell:

```
$ su - oracle
$ cat /proc/$$/mapped_base; echo
1073741824
$ sudo /usr/local/bin/ChangeMappedBase
$ cat /proc/$$/mapped_base; echo
268435456$
```

To change the mapping for shared libraries automatically during the Oracle log in process, execute:

```
# echo "sudo /usr/local/bin/ChangeMappedBase" >> ~/.bash_profile
```

Now log in as **oracle**:

```
$ ssh oracle@localhost
oracle@localhost's password:
Last login: Sun Jan  7 13:59:22 2003 from localhost
$ cat /proc/$$/mapped_base; echo
268435456$
```

Note

If the mapped base address for shared libraries for the Oracle processes was changed, then every Linux shell that spawns Oracle processes (for example, **listener**, **sqlplus**, etc.) must have the same mapped base address as well. If you execute **sqlplus** to connect to the local database, then you will get the following error message, seen in the screen below, if the **mapped_base** for this shell is not the same as for the running Oracle processes.

```
SQL> connect scott/tiger
ERROR:
ORA-01034: ORACLE not available
ORA-27102: out of memory
Linux Error: 12: Cannot allocate memory
Additional information: 1
Additional information: 491524

SQL>
```

Growing the Oracle SGA to 2.7/3.42 GB in x86 Red Hat Enterprise Linux 3, 4 and 5 Without VLM

16.1. General

Due to 32 bit virtual address limitations workarounds have been implemented in Linux to increase the maximum size for shared memories. A workaround is to lower the Mapped Base Address for shared libraries and the SGA Attach Address for shared memory segments. This enables Oracle to attain an SGA larger than 1.7 GB. To get a better understanding of address mappings in Linux and what Mapped Base Address is, see *Section 15.2, "Linux Memory Layout"*.

The following example shows how to increase the size of the SGA without a shared memory file system. A shared memory file system must be used on x86 to increase SGA beyond 3.42 GB, see *Section 14.8, "Huge Pages and Shared Memory File System in Red Hat Enterprise Linux 3"*.

16.2. Mapped Base Address for Shared Libraries in Red Hat Enterprise Linux 3, 4 and 5

In Red Hat Enterprise Linux 3, 4 or 5 the mapped base for shared libraries does not need to be lowered since this operation is now done automatically. To verify the mapped base (**`mapped_base`**) for shared libraries execute **`cat /proc/self/maps`** in a shell. The directory **`self`** in the proc file system always points to the current running process which in this example is the cat process:

```
# cat /etc/redhat-release
Red Hat Enterprise Linux AS release 3 (Taroon Update 6)
# cat /proc/self/maps
00a23000-00a38000 r-xp 00000000 08:09 14930   /lib/ld-2.3.2.so
00a38000-00a39000 rw-p 00015000 08:09 14930   /lib/ld-2.3.2.so
00b33000-00c66000 r-xp 00000000 08:09 69576   /lib/tls/libc-2.3.2.so
00c66000-00c69000 rw-p 00132000 08:09 69576   /lib/tls/libc-2.3.2.so
00c69000-00c6c000 rw-p 00000000 00:00 0
00ee5000-00ee6000 r-xp 00000000 08:09 32532   /etc/libcwait.so
00ee6000-00ee7000 rw-p 00000000 08:09 32532   /etc/libcwait.so
08048000-0804c000 r-xp 00000000 08:09 49318   /bin/cat
0804c000-0804d000 rw-p 00003000 08:09 49318   /bin/cat
099db000-099fc000 rw-p 00000000 00:00 0
b73e7000-b75e7000 r--p 00000000 08:02 313698  /usr/lib/locale/locale-archive
b75e7000-b75e8000 rw-p 00000000 00:00 0
bfff8000-c0000000 rw-p ffffc000 00:00 0
#

# cat /etc/redhat-release
Red Hat Enterprise Linux AS release 4 (Nahant Update 2)
# cat /proc/self/maps
00b68000-00b7d000 r-xp 00000000 03:45 1873128  /lib/ld-2.3.4.so
00b7d000-00b7e000 r--p 00015000 03:45 1873128  /lib/ld-2.3.4.so
00b7e000-00b7f000 rw-p 00016000 03:45 1873128  /lib/ld-2.3.4.so
00b81000-00ca5000 r-xp 00000000 03:45 1938273  /lib/tls/libc-2.3.4.so
00ca5000-00ca6000 r--p 00124000 03:45 1938273  /lib/tls/libc-2.3.4.so
00ca6000-00ca9000 rw-p 00125000 03:45 1938273  /lib/tls/libc-2.3.4.so
00ca9000-00cab000 rw-p 00ca9000 00:00 0
```

```
08048000-0804c000 r-xp 00000000 03:45 1531117  /bin/cat
0804c000-0804d000 rw-p 00003000 03:45 1531117  /bin/cat
08fa0000-08fc1000 rw-p 08fa0000 00:00 0
b7df9000-b7ff9000 r--p 00000000 03:45 68493    /usr/lib/locale/locale-archive
b7ff9000-b7ffa000 rw-p b7ff9000 00:00 0
bffa6000-c0000000 rw-p bffa6000 00:00 0
ffffe000-fffff000 ---p 00000000 00:00 0
#
```

The outputs show that the mapped base is already very low in Red Hat Enterprise Linux 3, 4 or 5. In the above example shared libraries start at 0xa38000 (decimal 10715136) in Red Hat Enterprise Linux 3 and 0xb68000 (decimal 11960320) in Red Hat Enterprise Linux 4 and 5. This is much lower than 0x40000000 (decimal 1073741824) in Red Hat Enterprise Linux 2.1:

```
# cat /etc/redhat-release
Red Hat Linux Advanced Server release 2.1AS (Pensacola)
# cat /proc/self/maps
08048000-0804c000 r-xp 00000000 08:08 44885      /bin/cat
0804c000-0804d000 rw-p 00003000 08:08 44885      /bin/cat
0804d000-0804f000 rwxp 00000000 00:00 0
40000000-40016000 r-xp 00000000 08:08 44751      /lib/ld-2.2.4.so
40016000-40017000 rw-p 00015000 08:08 44751      /lib/ld-2.2.4.so
40017000-40018000 rw-p 00000000 00:00 0
40022000-40155000 r-xp 00000000 08:08 47419      /lib/i686/libc-2.2.4.so
40155000-4015a000 rw-p 00132000 08:08 47419      /lib/i686/libc-2.2.4.so
4015a000-4015f000 rw-p 00000000 00:00 0
bffea000-bffee000 rwxp ffffd000 00:00 0
#
```

The above mappings show that the Mapped Base Address does not have to be lowered in Red Hat Enterprise Linux 3 or 4 to gain more SGA space.

16.3. Oracle 10g SGA Sizes in Red Hat Enterprise Linux 3, 4 or 5

The following table shows how large the Oracle 10g SGA can be configured in Red Hat Enterprise Linux 3, 4 or 5 without using a shared memory file system. Shared memory file systems for the SGA are covered at *Section 14.8, "Huge Pages and Shared Memory File System in Red Hat Enterprise Linux 3"*Configuring Very Large Memory (VLM).

Red Hat Enterprise Kernel Type	10g Database Version	Default Supported SGA without VLM	Max Supported SGA without VLM	Comments
smp kernel (x86)	10g Release 1	Up to 1.7 GB	Up to 2.7 GB	10g R1 must be relinked to increase the SGA size to approx 2.7 GB
hugemem kernel (x86)	10g Release 1	Up to 2.7 GB	Up to 3.42 GB	10g R1 must be relinked to increase the SGA size to approx 3.42 GB

Red Hat Enterprise Kernel Type	10g Database Version	Default Supported SGA without VLM	Max Supported SGA without VLM	Comments
smp kernel (x86)	10g Release 2	Up to ~2.2 GB (*)	Up to ~2.2 GB (*)	No relink of 10g R2 is necessary but the SGA Attach Address is a little bit higher than in R1
hugemem kernel (x86)	10g Release 2	Up to ~3.3 GB (*)	Up to ~3.3 GB (*)	No relink of 10g R2 is necessary but the SGA Attach Address is a little bit higher than in R1

Table 16.1. Table showing how large SGA can be configured in Red Hat Enterprise Linux

(*) When performing test scenarios with 10g R2 the database was not able to start up if sga_target was larger than 2350000000 bytes on a smp kernel, and if sga_target was larger than 3550000000 bytes on a hugemem kernel.

In Oracle 10g R2 the SGA size can be increased to approximately 2.7 GB using the smp kernel and to approximately 3.42 GB using the hugemem kernel. The SGA attach address does not have to be changed for that. To accommodate the same SGA sizes in Oracle 10g R1, the *Section 16.4, "Lowering the SGA Attach Address in Oracle 10g"* must be lowered.

Note

Lowering the SGA attach address in Oracle restricts the remaining 32 bit address space for Oracle processes. This means that less address space will be available for e.g. PGA memory. If the application uses a lot of PGA memory, then PGA allocations could fail even if there is sufficient free physical memory. Therefore, in certain cases it may be prudent not to change the SGA Attach Address to increase the SGA size but to use *Chapter 17, Using Very Large Memory (VLM)* instead. Also, if the SGA size is larger but less than 4GB to fit in memory address space, then the *Chapter 17, Using Very Large Memory (VLM)* solution should be considered first before switching to the hugemem kernel on a small system, unless the system has lots of physical memory. The hugemem kernel is not recommended on systems with less than 8GB of RAM due to some overhead issues in the kernel, see also *Section 2.2, "32 bit Architecture and the hugemem Kernel"*. If larger SGA sizes are needed than listed in the above table, then *Chapter 17, Using Very Large Memory (VLM)* must obviously be used on x86 platforms.

16.4. Lowering the SGA Attach Address in Oracle 10g

Starting with Oracle 10g R2 the SGA attach address does not have to be lowered for creating larger SGAs. However, Oracle 10g R1 must be relinked for larger SGAs.

The following commands were executed on a 10g R1 database system:

```
# ps -ef | grep "[o]ra_ckpt"
oracle    3035     1  0 23:21 ?        00:00:00 ora_ckpt_orcl
```

```
# cat /proc/3035/maps | grep SYSV
50000000-aa200000 rw-s 00000000 00:04 262144     /SYSV8b1d1510 (deleted)
#
```

The following commands were executed on a 10g R2 database system:

```
# ps -ef | grep "[o]ra_ckpt"
oracle    4998     1  0 22:29 ?        00:00:00 ora_ckpt_orcl
# cat /proc/4998/maps | grep SYSV
20000000-f4200000 rw-s 00000000 00:04 4390912    /SYSV950d1f70 (deleted)
#
```

The output shows that the SGA attach address in 10g R2 is already lowered to 0x20000000 vs. 0x50000000 in 10g R1. This means that Oracle 10g R2 does not have to be relinked for creating larger SGAs. For 10g R1 the SGA attach address must be lowered from 0x50000000 to e.g. 0xe000000. You could also set it a little bit higher like 0x20000000 as its done by default in 10g Release 2.

The following example shows how to lower the SGA attach address to 0xe000000 in 10g R1 (see also Metalink Note:329378.1):

```
su - oracle
cd $ORACLE_HOME/rdbms/lib
[[ ! -f ksms.s_orig ]] && cp ksms.s ksms.s_orig
genksms -s 0Xe000000 > ksms.s
make -f ins_rdbms.mk ksms.o
make -f ins_rdbms.mk ioracle
```

For a detailed description of these commands, see Lowering the SGA Attach Address for Shared Memory Segments in Oracle 9i.

You can verify the new lowered SGA attach address by running the following command:

```
$ objdump -t $ORACLE_HOME/bin/oracle |grep sgabeg
0e000000 l       *ABS*  00000000              sgabeg
$
```

Now when 10g R1 is restarted the SGA attach address should be at 0xe000000:

```
# ps -ef | grep "[o]ra_ckpt"
oracle    4998     1  0 22:29 ?        00:00:00 ora_ckpt_orcl
# cat /proc/4998/maps | grep SYSV
0e000000-c1200000 rw-s 00000000 00:04 0          /SYSV8b1d1510 (deleted)
#
```

Now you should be able to create larger SGAs.

Note

If you increase the size of the SGA, essentially using more process address space for the SGA, then less address space will be available for PGA memory. This means that if your application uses a lot of PGA memory, PGA allocations could fail even if you have sufficient RAM. In this case, you need to set the SGA attach address to a higher value which will lower the SGA size.

Using Very Large Memory (VLM)

17.1. General

This chapter does not apply to x86-64 (64 bit) systems.

With hugemem kernels on 32 bit systems, the SGA size can be increased but not significantly as shown at *Section 16.3, "Oracle 10g SGA Sizes in Red Hat Enterprise Linux 3, 4 or 5"*. Note the hugemem kernel is always recommended on systems with large amounts of RAM, see *Section 2.2, "32 bit Architecture and the hugemem Kernel"*. This chapter shows how the SGA can be significantly increased using VLM on 32 bit systems.

Starting with **Oracle9i Release 2** the SGA can theoretically be increased to about 62 GB (depending on block size) on a 32 bit system with 64 GB RAM. A processor feature called Page Address Extension (PAE) provides the capability of physically addressing 64 GB of RAM. However, it does not enable a process or program to address more than 4GB directly or have a virtual address space larger than 4GB. Hence, a process cannot attach to shared memory directly if it has a size of 4GB or more. To address this issue, a shared memory file system (memory-based file system) can be created which can be as large as the maximum allowable virtual memory supported by the kernel. With a shared memory file system processes can dynamically attach to regions of the file system allowing applications like Oracle to have virtually a much larger shared memory on 32 bit systems. *This is not an issue on 64 bit systems*.

For Oracle to use a shared memory file system, a feature called Very Large Memory (VLM) must be enabled. VLM moves the database buffer cache part of the SGA from the System V shared memory to the shared memory file system. It is still considered one large SGA but it consists now of two different OS shared memory entities. It is noteworthy to say that VLM uses 512MB of the non-buffer cache SGA to manage VLM. This memory area is needed for mapping the indirect data buffers (shared memory file system buffers) into the process address space since a process cannot attach to more than 4GB directly on a 32 bit system. For example, if the non-buffer cache SGA is 2.5 GB, then you will only have 2 GB of non-buffer cache SGA for shared pool, large pool, and redo log buffer since 512MB is used for managing VLM. If the buffer cache is less than 512 MB, then the **`init.ora`** parameter **`VLM_WINDOW_SIZE`** must be changed to reflect the size of the database buffer cache. However, it is not recommended to use VLM if **`db_block_buffers`** is not greater than 512MB.

In Red Hat Enterprise Linux 3, 4 and 5 there are two different memory file systems that can be used for VLM:

- **`shmfs`**/**`tmpfs`**: This memory file system is pageable and swappable and cannot be backed by Huge Pages because Huge Pages are not swappable.
- **`ramfs`**: This memory file systems is not pageable or swappable and not backed by Huge Pages, see also Huge Pages and Shared Memory File System in Red Hat Enterprise Linux 3 and 4.

Note that the **`shmfs`** file system is available in Red Hat Enterprise Linux 3 but not in Red Hat Enterprise Linux 4 or 5:

```
$ cat /etc/redhat-release
Red Hat Enterprise Linux AS release 3 (Taroon Update 6)
$ egrep "shm|tmpfs|ramfs" /proc/filesystems
nodev   tmpfs
nodev   shm
nodev   ramfs
$
```

```
$ cat /etc/redhat-release
Red Hat Enterprise Linux AS release 4 (Nahant Update 2)
$ egrep "shm|tmpfs|ramfs" /proc/filesystems
nodev   tmpfs
nodev   ramfs
$
```

This means that if you try to mount a **shmfs** file system in Red Hat Enterprise Linux 4 or 5, you will get the following error message:

```
mount: fs type shm not supported by kernel
```

The difference between **shmfs** and **tmpfs** is you do not need to specify the size of the file system if you **`mount`** a **tmpfs** file system.

17.2. Configuring Very Large Memory (VLM)

The following example shows how to use the RAM disk **ramfs** to allocate 8 GB of shared memory for the **Oracle 10g database** buffer cache on a 32 bit Red Hat Enterprise Linux 3, 4 or 5 systems (hugemem kernel). If this setup is performed on a server that does not have enough RAM, then Linux will appear to hang and the kernel will automatically start killing processes due to memory shortage (**`ramfs`** is not swappable). Furthermore, **`ramfs`** is not backed by Huge Pages and therefore the Huge Pages pool should not be increased for database buffers, see *Chapter 6, Swap Space*. In fact, if there are too many Huge Pages allocated, then there may not be enough memory for **ramfs**.

Since **ramfs** is not swappable, it is by default only usable by root. If you put too much on a ramfs file system, you can easily hang the system. To mount the **ramfs** file system and to make it usable for the Oracle account, execute:

```
# umount /dev/shm
# mount -t ramfs ramfs /dev/shm
# chown oracle:dba /dev/shm
```

When Oracle starts it will create a file in the **`/dev/shm`** directory that corresponds to the extended buffer cache. Ensure to add the above lines to **`/etc/rc.local`**. If **`ointall`** is the primary group of the Oracle account, use **`chown oracle:oinstall /dev/shm`** instead. For security reasons you do not want to give anyone write access to the shared memory file system. Having write access to the **`ramfs`** file system allows you to allocate and pin a large chunk of memory in RAM. In fact, you can kill a machine by allocating too much memory in the **`ramfs`** file system.

To enable VLM, set the Oracle parameter **`use_indirect_data_buffers`** to true:

```
use_indirect_data_buffers=true
```

For 10g R1 and R2 databases it is important to convert **`DB_CACHE_SIZE`** and **`DB_xK_CACHE_SIZE`** parameters to **`DB_BLOCK_BUFFERS`**, and to remove **`SGA_TARGET`** if set. Otherwise you will get errors like these:

```
ORA-00385: cannot enable Very Large Memory with new buffer cache parameters
```

Here is an example for configuring a 8 GB buffer cache for a 10g R2 database with Red Hat Enterprise Linux 3, 4 or 5 **`hugemem`** kernels:

```
use_indirect_data_buffers=true
db_block_size=8192
db_block_buffers=1048576
shared_pool_size=2831155200
```

Note that **shmmax** needs to be increased for **shared_pool_size** to fit into the System V shared memory. In fact, it should be slightly larger than the SGA size. Since **shared_pool_size** is less than 3 GB in this example, **shmmax** does not need to be larger than 3GB. The 8 GB indirect buffer cache will be in the RAM disk and hence it does not have to be accounted for in **shmmax**. On a 32 bit system the **shmmax** kernel parameter cannot be larger than 4GB, see also *Section 7.3, "Setting SHMALL Parameter"*.

In order to allow oracle processes to lock more memory into its address space for the VLM window size, the **ulimit** parameter **memlock** must be changed for oracle. Ensure to set **memlock** in **/etc/security/limits.conf** to 3145728:

```
oracle           soft    memlock         3145728
oracle           hard    memlock         3145728
Login as Oracle again and check max locked memory limit:
$ ulimit -l
3145728
```

If it is not working after a **ssh** log in, then you may have to set the SSH parameter **UsePrivilegeSeparation**, see *Chapter 22, Setting Shell Limits for Your Oracle User*.

If **memlock** is not set or too small, you will get error messages similar to this one:

```
ORA-27103: internal error
Linux Error: 11: Resource temporarily unavailable
```

Now try to start the database. Note the database start up can take a while. Also, the **sqlplus** banner or show sga may not accurately reflect the actual SGA size in older Oracle versions.

The 8GB file for the database buffer cache can be seen in the ramfs shared memory file system:

```
$ ls -al /dev/shm
total 120
drwxr-xr-x    1 oracle   dba             0    Nov 20 16:29  .
drwxr-xr-x   22 root     root       118784    Nov 20 16:25  ..
-rw-r-----    1 oracle   dba    8589934592    Nov 20 16:30  ora_orcl_458754
$
```

If the shared pool size is configured too large, you will get error messages similar to this one:

```
ORA-27103: internal error
Linux Error: 12: Cannot allocate memory
```

Part II. Installing the Oracle Database 10g on Red Hat Enterprise Linux

Introduction and Information on Supported Setups for Oracle 10g

This article provides a step by step guide for installing Oracle Database 10g on Red Hat Enterprise Linux. This guide covers the following Oracle Database and Red Hat Linux versions:

Oracle Database Version	Red Hat Enterprise Linux Version	Architecture	Comments
Oracle 10g R2 (10.2.0.1.0)	4 and 5	x86-64	See also *Oracle Database Installation Guide 10g Release 2 (10.2) for Linux x86-64*[1].
Oracle 10g R2 (10.2.0.1.0)	3, 4 and 5	x86	See also *Oracle Database Release Notes 10g Release 2 (10.2) for Linux x86*[2].
Oracle 10g R1 (10.1.0.3)	4	x86-64	See also *Oracle Database Installation Guide 10g Release 1 (10.1.0.3) for Linux x86-64*[3]
Oracle 10g R1 (10.1.0.3)	4	x86	See also *Oracle Database Release Notes 10g Release 1 (10.1.0.3.0) for Linux x86*[4].
Oracle 10g R1 (10.1.0.3)	3	x86-64	See also *Oracle Database Installation Guide 10g Release 1 (10.1.0.3) for Linux x86-64*[5].
Oracle 10g R1 (10.1.0.3)	3	x86	See also *Oracle Database Installation Guide 10g Release 1 (10.1) for UNIX Systems*[6].
Oracle 10g R1 (10.1.0.2)	2.1	x86	See also *Oracle Database Installation Guide 10g Release*

Oracle Database Version	Red Hat Enterprise Linux Version	Architecture	Comments
			1 (10.1) for UNIX Systems[7].

Table 3. Table displaying Certified Red Hat Enterprise Linux For Oracle 10g

For Validation and Certification information, see *Oracle's Certification Matrices*[8].

[8] https://metalink.oracle.com/metalink/certify/certify.welcome

Downloading and Unpacking Oracle 10g Installation Files

Download Oracle 10g (32 bit and 64 bit) for Linux from the following web site: *http://otn.oracle.com/software/products/database/oracle10g/index.html*

Note

To install an Oracle Database 10g (without RAC) you need to download the database file **`ship.db.lnx32.cpio.gz`**, **`10201_database_linux_x86_64.cpio`** or similar file.

Compute a cyclic redundancy check (CRC) checksum for the downloaded files and compare the checksum numbers against the numbers posted on the OTN website (where you downloaded the file from):

```
cksum ship.db.lnx32.cpio.gz
```

Uncompress the downloaded files:

```
gunzip ship.db.lnx32.cpio.gz
```

Unpack **`ship.db.lnx32.cpio`**:

```
$ cpio -idmv < ship.db.lnx32.cpio
Disk1/stage/Components/oracle.server/10.1.0.3.0/1
Disk1/stage/Components/oracle.server/10.1.0.3.0
Disk1/stage/Components/oracle.server
Disk1/stage/Components/oracle.tg/10.1.0.3.0/1/DataFiles
Disk1/stage/Components/oracle.tg/10.1.0.3.0/1
Disk1/stage/Components/oracle.tg/10.1.0.3.0
Disk1/stage/Components/oracle.tg
Disk1/stage/Components/oracle.assistants.dbca/10.1.0.3.0/1/DataFiles/doc.3.1.jar
Disk1/stage/Components/oracle.assistants.dbca/10.1.0.3.0/1/DataFiles/class.jar
...
```

Pre-Installation Preparation and Verification

19.1. Verifying Memory and Swap Space

Oracle states that the system must have at least 512MB of RAM and 1GB of swap space or a swap space twice the size of the RAM. For systems with more than 2 GB of RAM it states the swap space can be between one and two times the size of the RAM. You might also want to read Swap Space.

To check the size of physical memory, execute:

```
grep MemTotal /proc/meminfo
```

To check the size of swap space, execute:

```
grep SwapTotal /proc/meminfo
```

19.2. Verifying Temporary(`/tmp`) Space

According to Oracle's documentation, the Oracle Universal Installer (**`OUI`**) requires up to 400 MB of free space in the **`/tmp`** directory.

To check the space in **`/tmp`**, run:

```
$ df /tmp
```

If you do not have enough space in the **`/tmp`** file system, you can temporarily create a **`tmp`** directory in another file system. Here is how you can do this:

```
su - root
mkdir /<AnotherFilesystem> /tmp
chown root.root /<AnotherFilesystem> /tmp
chmod 1777 /<AnotherFilesystem> /tmp
export TEMP=/<AnotherFilesystem>
export TMPDIR=/<AnotherFilesystem>
```

The **`TEMP=/<AnotherFilesystem>`** file is used by Oracle. The TMPDIR=/<AnotherFilesystem> file is used by Linux programs like the linker "**`ld`**". Note as well <AnotherFilesystem> should be adjusted to fit your filesystem. When you are done with the Oracle installation, shut down Oracle and remove the temporary **`/tmp`** directory:

```
su - root
rmdir /<AnotherFilesystem>/tmp
unset TEMP
unset TMPDIR
```

19.3. Verifying Software Packages (RPMs)

Before you install an Oracle Database 10g you need to check your system for required RPMs. Always use the latest stable RPMs and kernels for your system, Red Hat manages the repositories to ensure

everything work safely, securely and accurately. This section contains a list Oracle Database versions and versions of Red Hat Enterprise Linux. The list details which packages are required and how to assess whether the correct RPMs are present.

For **Oracle 10g R2 (64 bit)**, on Red Hat Enterprise Linux 4 and 5 x86-64 versions, the document *Oracle Database Installation Guide 10g Release 2 (10.2) for Linux x86-64*[1] lists the following required package versions or higher:

```
binutils-2.15.92.0.2-10.EL4
compat-db-4.1.25-9
control-center-2.8.0-12
gcc-3.4.3-9.EL4
gcc-c++-3.4.3-9.EL4
glibc-2.3.4-2
glibc-common-2.3.4-2
gnome-libs-1.4.1.2.90-44.1
libstdc++-3.4.3-9.EL4
libstdc++-devel-3.4.3-9.EL4
make-3.80-5
pdksh-5.2.14-30
sysstat-5.0.5-1
xscreensaver-4.18-5.rhel4.2
```

Note

Install the **`libaio-0.3.96`** RPM or a newer version, otherwise the OUI prerequisite check will fail.

To check if you are running the x86-64 kernel on a x86-64 platform, run:

```
# uname -mi
x86_64 x86_64
```

To check the RPMs, run:

```
rpm -q --qf '%{NAME}-%{VERSION}-%{RELEASE}  (%{ARCH})\n' \
    binutils compat-db control-center gcc gcc-c++ glibc glibc-common \
    gnome-libs libstdc++ libstdc++-devel make pdksh sysstat xscreensaver \
    libaio
```

It is important to have these x86-64 RPMs installed. The above command will list the architecture of each binary package. You will see that some RPMs are installed twice when you run this command (x86 RPM and x86-64 RPM). You need to ensure that all required x86-64 RPMs listed here are installed.

For **Oracle 10g R2 (32 bit)** on Red Hat Enterprise Linux 4 and 5 x86, the document *Oracle Database Release Notes 10g Release 2 (10.2) for Linux x86*[2] lists the following required package versions or higher:

```
binutils-2.15.92.0.2-10.EL4
compat-db-4.1.25-9
control-center-2.8.0-12
gcc-3.4.3-9.EL4
```

[1] http://download-east.oracle.com/docs/cd/B19306_01/install.102/b15667/toc.htm
[2] http://download-east.oracle.com/docs/html/B15659_03/toc.htm

```
gcc-c++-3.4.3-9.EL4
glibc-2.3.4-2
glibc-common-2.3.4-2
gnome-libs-1.4.1.2.90-44.1
libstdc++-3.4.3-9.EL4
libstdc++-devel-3.4.3-9.EL4
make-3.80-5
pdksh-5.2.14-30
sysstat-5.0.5-1
xscreensaver-4.18-5.rhel4.2
```

Note

Install the **libaio-0.3.96** RPM or a newer version, otherwise the OUI prerequisite check will fail.

To check the RPMs, run:

```
rpm -q binutils compat-db control-center gcc gcc-c++ glibc glibc-common \
 gnome-libs libstdc++ libstdc++-devel make pdksh sysstat xscreensaver libaio
```

For 10g R2 (32 bit) on Red Hat Enterprise Linux 3 x86, the document Oracle Database Installation Guide 10g Release 2 (10.2) for Linux x86 lists the following required package versions or higher:

```
make-3.79.1
gcc-3.2.3-34
glibc-2.3.2-95.20
compat-db-4.0.14-5
compat-gcc-7.3-2.96.128
compat-gcc-c++-7.3-2.96.128
compat-libstdc++-7.3-2.96.128
compat-libstdc++-devel-7.3-2.96.128
openmotif21-2.1.30-8
setarch-1.3-1
```

Note

libaio-0.3.96 RPM or a newer version is required for the OUI to work successfully.

To check the RPMs, run:

```
rpm -q make gcc glibc compat-db compat-gcc compat-gcc-c++ compat-libstdc++ \
 compat-libstdc++-devel openmotif21 setarch libaio
```

For Oracle 10g R1 (64 bit) on Red Hat Enterprise Linux 3 x86-64, the document Oracle Database Installation Guide 10g Release 1 (10.1.0.3) for Linux x86-64 lists the following required package versions or higher:

```
make-3.79.1
gcc-3.2.3-34
glibc-2.3.2-95.20
glibc-devel-2.3.2-95.20
glibc-devel-2.3.2-95.20   (32 bit)
compat-db-4.0.14-5
```

```
compat-gcc-7.3-2.96.128
compat-gcc-c++-7.3-2.96.128
compat-libstdc++-7.3-2.96.128
compat-libstdc++-devel-7.3-2.96.128
gnome-libs-1.4.1.2.90-34.1   (32 bit)
openmotif21-2.1.30-8
setarch-1.3-1
libaio-0.3.96-3
libaio-devel-0.3.96-3
```

To check if you are running the AMD64 or Intel 64(x86-64) kernel on an AMD64 or Intel 64 platform, run:

```
# uname -mi
x86_64 x86_64
```

To check the RPMs, run:

```
rpm -q --qf '%{NAME}-%{VERSION}-%{RELEASE}  (%{ARCH})\n' \
make gcc glibc glibc-devel compat-db compat-gcc compat-gcc-c++ \
 compat-libstdc++ compat-libstdc++-devel gnome-libs openmotif21 setarch \
 libaio libaio-devel
```

It is important to have the right x86 and x86-64 RPMs installed. The above command will list the architecture of each binary package. And as you can see in the above list, glibc-devel and other RPMs are listed twice. This means that you have to install packages for both architectures, x86 and x86-64.

For Oracle 10g R1 (32 bit) on Red Hat Enterprise Linux 3 x86, the document Oracle Database Installation Guide 10g Release 1 (10.1) for UNIX Systems lists the following required package versions or higher:

```
make-3.79.1
gcc-3.2.3-34
glibc-2.3.2-95.20
compat-db-4.0.14-5
compat-gcc-7.3-2.96.128
compat-gcc-c++-7.3-2.96.128
compat-libstdc++-7.3-2.96.128
compat-libstdc++-devel-7.3-2.96.128
openmotif21-2.1.30-8
setarch-1.3-1
```

To check the RPMs, run:

```
rpm -q make gcc glibc compat-db compat-gcc compat-gcc-c++ compat-libstdc++ \
 compat-libstdc++-devel openmotif21 setarch
```

For Oracle 10g R1 (32 bit) on Red Hat Enterprise Linux 2.1, the document Oracle Database Installation Guide 10g Release 1 (10.1) for UNIX Systems lists the following required package versions or higher:

```
make-3.79.1
glibc-2.2.4-32
gcc-2.96-128
gcc-c++-2.96-128
libstdc++-2.96-128
openmotif-2.1.30-11
```

To check these RPMs, run:

```
rpm -q make glibc gcc gcc-c++ libstdc++ openmotif
```

For Red Hat Enterprise Linux 3 and 2.1 it is also important to have **binutils-2.11.90.0.8-12** or a newer version installed. Make sure you have the

```
binutils
```

RPM installed on Red Hat Enterprise Linux 4 and 5 as well:

```
rpm -q binutils
```

Important

OUI for x86 will also complain if the **openmotif** package is missing (do not confuse it with the **openmotif21** package). Red Hat changed the version naming schema from **openmotif-2.2.2-16** in the original release to **openmotif-2.2.3-5.RHEL3.2** in Red Hat Enterprise Linux 3 Update 5. This seems to confuse OUI in Red Hat Enterprise Linux3 U5 since it complaining that it can not find the right **openmotif** version. You can ignore this. The openmotif-2.2.3-5.RHEL3.2 is just a newer version of **openmotif-2.2.2-16** which should work fine and should not cause any problems. To check the RPM, run:

```
rpm -q openmotif
```

Also, make sure the redhat-release package is installed. Earlier versions of Red Hat Enterprise Linux may not install it by default when you selected a minimum system installation:

```
rpm -q redhat-release
```

The **setarch** utility is new in Red Hat Enterprise Linux 3 and 4. It is used to tell the kernel to report a different architecture than the current one. It is also used to emulate a 3GB virtual address space for applications that do not run properly with a larger virtual address space. To check the RPM, run:

```
rpm -q setarch
```

19.4. Verifying Kernel Parameters

To see all kernel parameters, execute:

```
su - root
sysctl -a
```

For Oracle 10g, the following kernel parameters have to be set to values greater than or equal to the recommended values which can be changed in the **proc** file system:

To verify **shmmax**, execute:

```
cat /proc/sys/kernel/shmmax
shmmax  = 2147483648
```

To verify **shmmni**, execute:

```
cat /proc/sys/kernel/shmmni
shmmni  = 4096
```

To verify the **shmall** parameter, execute the command below. **shmall** is used in 10g R1.

```
cat /proc/sys/kernel/shmall
shmall  = 2097152
```

To verify shmmin, execute:

```
ipcs -lm |grep "min seg size"
shmmin  = 1
```

Note that **shmseg** is hardcoded in the kernel, the default is much higher.

```
shmseg  = 10
```

To verify **semmsl**, execute:

```
cat /proc/sys/kernel/sem | awk '{print $1}'
semmsl  = 250
```

To verify **semmns**, execute:

```
cat /proc/sys/kernel/sem | awk '{print $2}'
semmns  = 32000
```

To verify **semopm**, execute:

```
cat /proc/sys/kernel/sem | awk '{print $3}'
semopm  = 100
```

To verify **semmni**, execute:

```
cat /proc/sys/kernel/sem | awk '{print $4}'
semmni  = 128
```

To verify **file-max**, execute:

```
cat /proc/sys/fs/file-max
file-max = 65536
```

To verify **ip_local_port_range**, execute:

```
cat /proc/sys/net/ipv4/ip_local_port_range
ip_local_port_range = 1024 65000
```

Note

Do not change the value of any kernel parameter on a system where it is already higher than listed as minimum requirement.

On the following versions of Red Hat Enterprise Linux 4 x86, 3 U5 x86, 3 U5 x86-64, and 2.1; you may have to increase the kernel parameters **shmmax**, **semopm**, and **filemax** to meet the minimum requirement. On Red Hat Enterprise Linux 4 x86-64 you may have to increase **shmmax** and **semopm**.

Oracle also recommends to set the local port range **ip_local_port_range** for outgoing messages to "1024 65000" which is needed for high usage systems. This kernel parameter defines the local port range for TCP and UDP traffic to choose from.

In order to meet these requirements, you may have to add the following lines to the **/etc/sysctl.conf** file which are read during the boot process:

```
kernel.shmmax=2147483648
kernel.sem=250 32000 100 128
fs.file-max=65536
net.ipv4.ip_local_port_range=1024 65000
```

Adding these lines to the **/etc/sysctl.conf** file will cause the system to change these kernel parameters after each boot using the **/etc/rc.d/rc.sysinit** script which is invoked by **/etc/inittab**. But in order that these new added lines or settings in **/etc/sysctl.conf** become effective immediately, execute the following command:

```
su - root
sysctl -p
```

For more information on shmmax, shmmni, shmmin, shmseg, and shmall, see *Chapter 7, Setting Shared Memory*. For more information on semmsl, semmni, semmns, and semopm, see *Chapter 24, Creating Oracle Directories*. For more information on filemax, see *Chapter 9, Setting File Handles*.

Starting with 10g R2 some network settings must be adjusted as well which is checked by OUI. Oracle recommends the default and maximum send buffer size (**SO_SNDBUF** socket option) and receive buffer size (**SO_RCVBUF** socket option) to be set to 256 KB. The receive buffers are used by TCP and UDP to hold the received data for the application until it is read. This buffer cannot overflow because the sending party is not allowed to send data beyond the buffer size window. This means that datagrams will be discarded if they do not fit in the receive buffer. This could cause the sender to overwhelm the receiver.

The default and maximum window size can be changed in the proc file system without reboot by running the following commands: The default setting in bytes of the socket receive buffer.

```
# sysctl -w net.core.rmem_default=262144
```

The default setting in bytes of the socket send buffer.

```
# sysctl -w net.core.wmem_default=262144
```

The maximum socket receive buffer size which may be set by using the SO_RCVBUF socket option.

```
# sysctl -w net.core.rmem_max=262144
```

The maximum socket send buffer size which may be set by using the SO_SNDBUF socket option.

```
# sysctl -w net.core.wmem_max=262144
```

To make the change permanent, add the following lines to the **/etc/sysctl.conf** file, which is used during the boot process:

```
net.core.rmem_default=262144
net.core.wmem_default=262144
net.core.rmem_max=262144
net.core.wmem_max=262144
```

Installing Required Software Packages

The following sections provide a detailed set of instructions for installing the required software packages(RPMs) before you install your Oracle Database 10g (R1 and R2). These sections include the following possible configurations:

- 10g R2 on Red Hat Enterprise Linux 4 and 5 (x86-64),
- 10g R2 on Red Hat Enterprise Linux 4 and 5 (x86),
- 10g R1 on Red Hat Enterprise Linux 4 and 5 (x86-64),
- 10g R1 on Red Hat Enterprise Linux 4 and 5 (x86),
- 10g R1 and R2 on Red Hat Enterprise Linux 3 (x86),
- 10g R1 on Red Hat Enterprise Linux 3 (x86-64), and
- 10g R1 on Red Hat Enterprise Linux 2.1 (x86).

20.1. 10g R2 on Red Hat Enterprise Linux 4 and 5 x86-64 version

On Red Hat Enterprise Linux 4 x86-64 you may have to install the following RPMs and dependencies:

```
# rpm -Uvh gcc-3.4.4-2.x86_64.rpm      \
    gcc-c++-3.4.4-2.x86_64.rpm                  \
    libstdc++-devel-3.4.4-2.x86_64.rpm          \
    cpp-3.4.4-2.x86_64.rpm                      \
    glibc-devel-2.3.4-2.13.x86_64.rpm           \
    glibc-headers-2.3.4-2.13.x86_64.rpm         \
    glibc-kernheaders-2.4-9.1.98.EL.x86_64.rpm

#rpm -Uvh gnome-libs-1.4.1.2.90-44.1.x86_64.rpm   \
    compat-db-4.1.25-9.x86_64.rpm                   \
    ORBit-0.5.17-14.x86_64.rpm                      \
    gtk+-1.2.10-33.x86_64.rpm                       \
    imlib-1.9.13-23.x86_64.rpm                      \
    libpng10-1.0.16-1.x86_64.rpm                    \
    gdk-pixbuf-0.22.0-16.el4.x86_64.rpm             \
    libungif-4.1.3-1.x86_64.rpm

# rpm -Uvh sysstat-5.0.5-1.x86_64.rpm
```

You also need to install the following i386 and x86-64 RPMs if not already installed, otherwise you will get various different error messages.

For a detailed list of error messages, read *Chapter 29, Oracle 10g and Linux Installation Errors and Troubleshooting*.

```
rpm -Uvh xorg-x11-deprecated-libs-6.8.2-1.EL.13.20.i386.rpm \
    xorg-x11-libs-6.8.2-1.EL.13.20.i386.rpm                     \
    xorg-x11-Mesa-libGL-6.8.2-1.EL.13.20.i386.rpm               \
    expat-1.95.7-4.i386.rpm                                     \
    fontconfig-2.2.3-7.i386.rpm                                 \
    freetype-2.1.9-1.i386.rpm                                   \
    zlib-1.2.1.2-1.2.i386.rpm
```

```
rpm -Uvh libaio-0.3.103-3.x86_64.rpm

rpm -Uvh compat-libstdc++-33-3.2.3-47.3.x86_64.rpm

rpm -Uvh glibc-devel-2.3.4-2.13.i386.rpm libgcc-3.4.4-2.i386.rpm
```

If you are yet to install Update 3 or later, do not forget to install an updated **`binutils`** RPM from *https://rhn.redhat.com/* or from *http://oss.oracle.com/*[1]. This command will install the newer version of **`binutils`**, remember to change the version number if the one you downloaded is newer.

```
rpm -Uvh --force binutils-2.15.92.0.2-13.0.0.0.2.x86_64.rpm
```

If you do not install a newer binutil RPM from Oracle or RHN, then you will get the following error message:

```
/usr/bin/ld: /u01/app/oracle/oracle/product/10.2.0/db_1/lib//libirc.a(fast_memcpy.o):
    relocation R_X86_64_PC32 against `_memcpy_mem_ops_method' can not
    be used when making a shared object; recompile with -fPIC
    /usr/bin/ld: final link failed: Bad value
    collect2: ld returned 1 exit status
```

For more information on this bug, see *Bugzilla Bug 679*[2].

Oracle lists the **`control-center`** and **`xscreensaver`** RPMs as a requirements. But you should not have any problems when these RPMs are missing. But if you want to install them, you may have to install many additional RPMs in order to satisfy dependencies:

```
rpm -Uvh gcc-3.4.4-2.x86_64.rpm \
    gcc-c++-3.4.4-2.x86_64.rpm \
    libstdc++-devel-3.4.4-2.x86_64.rpm \
    cpp-3.4.4-2.x86_64.rpm \
    glibc-devel-2.3.4-2.13.x86_64.rpm \
    glibc-headers-2.3.4-2.13.x86_64.rpm \
    glibc-kernheaders-2.4-9.1.98.EL.x86_64.rpm

rpm -Uvh control-center-2.8.0-12.rhel4.2.x86_64.rpm \
    xscreensaver-4.18-5.rhel4.9.x86_64.rpm \
    eel2-2.8.1-2.x86_64.rpm \
    gail-1.8.0-2.x86_64.rpm \
    gnome-desktop-2.8.0-5.x86_64.rpm \
    gnome-icon-theme-2.8.0-1.el4.1.3.noarch.rpm \
    libgail-gnome-1.1.0-1.x86_64.rpm \
    libxklavier-1.02-3.x86_64.rpm \
    metacity-2.8.6-2.8.x86_64.rpm \
    nautilus-2.8.1-4.x86_64.rpm \
    startup-notification-0.7-1.x86_64.rpm \
    xloadimage-4.1-34.RHEL4.x86_64.rpm \
    xorg-x11-Mesa-libGLU-6.8.2-1.EL.13.20.x86_64.rpm \
    at-spi-1.6.0-3.x86_64.rpm \
    desktop-backgrounds-basic-2.0-26.2.1E.noarch.rpm \
    eog-2.8.1-2.x86_64.rpm \
    gnome-panel-2.8.1-3.3E.x86_64.rpm \
    gnome-vfs2-smb-2.8.2-8.2.x86_64.rpm \
    hicolor-icon-theme-0.3-3.noarch.rpm \
    libexif-0.5.12-5.1.x86_64.rpm \
    librsvg2-2.8.1-1.x86_64.rpm \
    nautilus-cd-burner-2.8.3-6.x86_64.rpm \
```

[1] http://oss.oracle.com/projects/compat-oracle/files/RedHat/
[2] http://sources.redhat.com/bugzilla/show_bug.cgi?id=679

```
    redhat-artwork-0.120.1-1.2E.x86_64.rpm \
    scrollkeeper-0.3.14-3.x86_64.rpm \
    cdrecord-2.01.1-5.x86_64.rpm \
    docbook-dtds-1.0-25.noarch.rpm \
    evolution-data-server-1.0.2-9.x86_64.rpm \
    intltool-0.31.2-1.x86_64.rpm \
    libcroco-0.6.0-4.x86_64.rpm \
    libgnomeprint22-2.8.0-3.x86_64.rpm \
    libgnomeprintui22-2.8.0-1.x86_64.rpm \
    libgsf-1.10.1-1.x86_64.rpm \
    libwnck-2.8.1-1.rhel4.1.x86_64.rpm \
    mkisofs-2.01.1-5.x86_64.rpm \
    samba-common-3.0.10-1.4E.2.x86_64.rpm \
    ghostscript-7.07-33.x86_64.rpm \
    ghostscript-fonts-5.50-13.noarch.rpm \
    gnutls-1.0.20-3.2.1.x86_64.rpm \
    libgnomecups-0.1.12-5.x86_64.rpm \
    libsoup-2.2.1-2.x86_64.rpm \
    openjade-1.3.2-14.x86_64.rpm \
    perl-XML-Parser-2.34-5.x86_64.rpm \
    sgml-common-0.6.3-17.noarch.rpm \
    urw-fonts-2.2-6.1.noarch.rpm \
    xml-common-0.6.3-17.noarch.rpm \
    VFlib2-2.25.6-25.x86_64.rpm \
    chkfontpath-1.10.0-2.x86_64.rpm \
    perl-URI-1.30-4.noarch.rpm \
    perl-libwww-perl-5.79-5.noarch.rpm \
    xorg-x11-font-utils-6.8.2-1.EL.13.20.x86_64.rpm \
    perl-HTML-Parser-3.35-6.x86_64.rpm \
    xorg-x11-xfs-6.8.2-1.EL.13.20.x86_64.rpm \
    perl-HTML-Tagset-3.03-30.noarch.rpm \
    ttmkfdir-3.0.9-14.1.EL.x86_64.rpm
```

20.2. 10g R2 on Red Hat Enterprise Linux 4 and 5 (x86)

On Red Hat Enterprise Linux 4 and 5 x86 you may have to install the following RPMs and dependencies to meet the software requirements:

```
rpm -Uvh gcc-3.4.4-2.i386.rpm \
    gcc-c++-3.4.4-2.i386.rpm \
    libstdc++-devel-3.4.4-2.i386.rpm \
    glibc-devel-2.3.4-2.13.i386.rpm \
    glibc-headers-2.3.4-2.13.i386.rpm \
    glibc-kernheaders-2.4-9.1.98.EL.i386.rpm

rpm -Uvh gnome-libs-1.4.1.2.90-44.1.i386.rpm \
    compat-db-4.1.25-9.i386.rpm \
    ORBit-0.5.17-14.i386.rpm \
    gtk+-1.2.10-33.i386.rpm \
    imlib-1.9.13-23.i386.rpm \
    libpng10-1.0.16-1.i386.rpm \
    gdk-pixbuf-0.22.0-16.el4.i386.rpm \
    libungif-4.1.3-1.i386.rpm \
    alsa-lib-1.0.6-5.RHEL4.i386.rpm \
    audiofile-0.2.6-1.i386.rpm \
    esound-0.2.35-2.i386.rpm

rpm -Uvh sysstat-5.0.5-1.i386.rpm

rpm -Uvh libaio-0.3.103-3.i386.rpm

rpm -Uvh xorg-x11-deprecated-libs-6.8.2-1.EL.13.20.i386.rpm
```

```
rpm -Uvh compat-libstdc++-33-3.2.3-47.3.i386.rpm
```

Oracle lists the **control-center** and **xscreensaver** RPMs as a requirements. But you should not have any problems when these RPMs are missing. If you want to install them, you may have to install many additional RPMs in order to satisfy dependencies.

20.3. 10g R1 on Red Hat Enterprise Linux 4 and 5 (x86-64)

On Red Hat Enterprise Linux 4 x86-64 you may have to install the following RPMs and dependencies:

```
rpm -Uvh gcc-3.4.3-22.1.x86_64.rpm \
    cpp-3.4.3-22.1.x86_64.rpm \
    glibc-devel-2.3.4-2.9.x86_64.rpm \
    glibc-headers-2.3.4-2.9.x86_64.rpm \
    glibc-kernheaders-2.4-9.1.87.x86_64.rpm

rpm -Uvh glibc-devel-2.3.4-2.9.i386.rpm

rpm -Uvh openmotif-2.2.3-9.RHEL4.1.x86_64.rpm \
    xorg-x11-deprecated-libs-6.8.2-1.EL.13.6.x86_64.rpm

rpm -Uvh xorg-x11-deprecated-libs-6.8.2-1.EL.13.6.i386.rpm \
    xorg-x11-libs-6.8.2-1.EL.13.6.i386.rpm \
    xorg-x11-Mesa-libGL-6.8.2-1.EL.13.6.i386.rpm \
    expat-1.95.7-4.i386.rpm fontconfig-2.2.3-7.i386.rpm \
    freetype-2.1.9-1.i386.rpm zlib-1.2.1.2-1.i386.rpm

rpm -Uvh libgcc-3.4.3-22.1.i386.rpm
```

20.4. 10g R1 on Red Hat Enterprise Linux 4 and 5 (x86)

On Red Hat Enterprise Linux 4 x86 you may have to install the following RPMs and dependencies to meet the software requirements:

```
rpm -Uvh gcc-3.4.3-9.EL4.i386.rpm \
    glibc-devel-2.3.4-2.i386.rpm \
    glibc-headers-2.3.4-2.i386.rpm \
    glibc-kernheaders-2.4-9.1.87.i386.rpm

rpm -Uvh openmotif-2.2.3-6.RHEL4.2.i386.rpm \
    xorg-x11-deprecated-libs-6.8.1-23.EL.i386.rpm
```

Note

The 10g 10.1.0.3 OUI Product specific prerequisite check will fail for the **gcc**, **binutils**, and **openmotif** versions. You can ignore these failed checks and proceed.

Important Note

The redhat-release RPM should already be installed by default. But note that 10.1.0.3.0 OUI does not recognize Red Hat Enterprise Linux 4 or 5 as a supported release yet. This means you will have to edit the **/etc/redhat-release** file, see below, or you apply the 4153257 patch for 10g R1 on Red Hat Enterprise Linux 4 and 5.

20.5. Oracle 10g R1 and R2 on Red Hat Enterprise Linux 3 (x86)

On Red Hat Enterprise Linux 3 x86 you may have to install the following RPMs and dependencies to meet the software requirements:

```
rpm -Uvh gcc-3.2.3-52.i386.rpm \
    cpp-3.2.3-52.i386.rpm \
    glibc-devel-2.3.2-95.33.i386.rpm \
    glibc-headers-2.3.2-95.33.i386.rpm \
    glibc-kernheaders-2.4-8.34.1.i386.rpm

rpm -Uvh compat-db-4.0.14-5.1.i386.rpm \
    compat-gcc-7.3-2.96.128.i386.rpm \
    compat-gcc-c++-7.3-2.96.128.i386.rpm \
    compat-libstdc++-7.3-2.96.128.i386.rpm \
    compat-libstdc++-devel-7.3-2.96.128.i386.rpm \
    tcl-8.3.5-92.2.i386.rpm

rpm -Uvh libaio-0.3.96-5.i386.rpm

rpm -Uvh openmotif21-2.1.30-9.RHEL3.6.i386.rpm

rpm -Uvh openmotif-2.2.3-5.RHEL3.2.i386.rpm
```

20.6. Oracle 10g R1 on Red Hat Enterprise Linux 3 (x86_64)

On Red Hat Enterprise Linux 3 x86_64 you may have to install the following RPMs and dependencies to meet the software requirements:

```
rpm -Uvh gcc-3.2.3-52.x86_64.rpm \
    cpp-3.2.3-52.x86_64.rpm \
    glibc-devel-2.3.2-95.33.x86_64.rpm \
    glibc-headers-2.3.2-95.33.x86_64.rpm \
    glibc-kernheaders-2.4-8.34.1.x86_64.rpm

rpm -Uvh glibc-devel-2.3.2-95.33.i386.rpm

rpm -Uvh compat-db-4.0.14-5.1.x86_64.rpm \
    compat-gcc-7.3-2.96.128.i386.rpm \
    compat-gcc-c++-7.3-2.96.128.i386.rpm \
    compat-libstdc++-7.3-2.96.128.i386.rpm \
    compat-libstdc++-devel-7.3-2.96.128.i386.rpm \
    tcl-8.3.5-92.2.x86_64.rpm \
    libgcc-3.2.3-52.i386.rpm

rpm -Uvh libaio-0.3.96-5.x86_64.rpm \
    libaio-devel-0.3.96-5.x86_64.rpm
```

Note: Red Hat Enterprise Linux 3 x86-64 U5 does not come with a i386 gnome-libs RPM

```
rpm -Uvh gnome-libs-1.4.1.2.90-34.2.x86_64.rpm \
    ORBit-0.5.17-10.4.x86_64.rpm \
    audiofile-0.2.3-7.1.x86_64.rpm \
    esound-0.2.28-6.x86_64.rpm \
    gtk+-1.2.10-31.x86_64.rpm \
    imlib-1.9.13-13.4.x86_64.rpm \
    gdk-pixbuf-0.22.0-12.el3.x86_64.rpm \
    libpng10-1.0.13-15.x86_64.rpm \
```

```
    libungif-4.1.0-15.x86_64.rpm
```

Note: Red Hat Enterprise Linux 3 x86-64 U5 does not come with a x86-64 openmotif21 RPM

```
rpm -Uvh openmotif21-2.1.30-9.RHEL3.6.i386.rpm \
    XFree86-libs-4.3.0-81.EL.i386.rpm \
    XFree86-Mesa-libGL-4.3.0-81.EL.i386.rpm \
    expat-1.95.5-6.i386.rpm \
    fontconfig-2.2.1-13.i386.rpm \
    freetype-2.1.4-4.0.i386.rpm \
    zlib-1.1.4-8.1.i386.rpm
```

Make sure to use the right i386 and x86_64 RPMs as listed above.

Note, if you do not install the i386 **`XFree86-libs`** RPM, you will get an error message similar to this one:

```
/tmp/OraInstall2005-06-15_07-36-25AM/jre/1.4.2/lib/i386/libawt.so: \
libXp.so.6: cannot open shared object file: No such file or directory
```

For more information, see *Chapter 29, Oracle 10g and Linux Installation Errors and Troubleshooting*.

20.7. Oracle 10g R1 on Red Hat Enterprise Linux 2.1 (x86)

On Red Hat Enterprise Linux 2.1 x86 you may have to install the following RPMs and dependencies to meet the software requirements:

```
rpm -Uvh glibc-2.2.4-32.11.i686.rpm \
    glibc-common-2.2.4-32.11.i386.rpm

rpm -Uvh gcc-2.96-108.1.i386.rpm \
    binutils-2.11.90.0.8-12.i386.rpm \
    cpp-2.96-108.1.i386.rpm \
    glibc-devel-2.2.4-32.11.i386.rpm \
    kernel-headers-2.4.9-e.3.i386.rpm

rpm -Uvh openmotif-2.1.30-11.i386.rpm

rpm -Uvh redhat-release-as-2.1AS-4.noarch.rpm
```

You may have to upgrade **`glibc`** in order to pass Oracle's "Product-specific Prerequisite" checks. Oracle's recommended **`glibc`** version is 2.2.4.31.7 or higher.

There is no setarch RPM for Red Hat Enterprise Linux 2.1.

Also, it is important to install a newer kernel version for Red Hat Enterprise Linux 2.1. Do not use a kernel older than 2.4.9-e.25. To check the kernel version run **`uname -r`**.

20.8. Verifying and Updating the redhat-release File

Verify that the redhat-release RPM is installed on your Red Hat system:

```
rpm -q redhat-release
```

This RPM is important for Red Hat Enterprise Linux since Red Hat Enterprise Linux 2.1, 3, 4 and 5 are Linux releases supported by Oracle. Without this RPM, Oracle 10g OUI will not be able to recognize

it as a supported OS. However, the installer of 10g 10.1.0.3 does not recognize Red Hat Enterprise Linux 4 as a supported release yet. This means that you will have to edit the **/etc/redhat-release** file. It is not recommend to execute "**runInstaller -ignoreSysPrereqs**" since this will disable other checks you may need.

On Red Hat Enterprise Linux 4 (for 10g R1) you have to change the **/etc/redhat-release** file to make Oracle 10g believe it is running on a supported release.

Regarding Red Hat Enterprise Linux 4, the installer for 10g 10.1.0.3 does not recognize Red Hat Enterprise Linux 4 as a supported release but 10g R2 OUI does.

To change the **/etc/redhat-release** file, you can simply copy and paste the following commands:

```
su - root
# cp /etc/redhat-release /etc/redhat-release.orig
# cat > /etc/redhat-release << EOF
Red Hat Enterprise Linux AS release 3 (Taroon)
EOF
```

After you are done with the Oracle 10g installation, undo the changes you made to **/etc/redhat-release**:

```
su - root
# cp /etc/redhat-release.orig /etc/redhat-release
```

Sizing Disk Space for Oracle 10g

Oracle says that about 2.5 GB of disk space should be reserved for the Oracle software on Linux.

Here are examples to check a file system:

```
$ du -m -s /u01
1963    /u01
$ du -m -s /u01/app/oracle/oradata
720     /u01/app/oracle/oradata
```

If you also install additional software from the Oracle Database 10g Companion CD, then add at least 1 GB of free disk space.

So if you install Oracle 10g Enterprise Edition and additional software from the Oracle Database 10g Companion CD, then you need about 2.5 GB of disk for the Oracle software. And if you also want to add a pre-configured database on the same file system, make sure to add another 1 GB of disk space.

Note

If you do not put Oracle 10g on a separate file systems, then make sure the root file system "**/**" has enough disk space. You can check the free space of the root file system with the following command:

```
df -h /
```

Setting Shell Limits for Your Oracle User

Most shells like Bash provide control over various resources like the maximum allowable number of open file descriptors or the maximum number of processes available to a user. For more information on **`ulimit`** for **`Bash`**, see **`man bash`** and search for **`ulimit`**.

If you just install a small test database, then you might be fine with the current settings. Please note that the limits very often vary from system to system. But for larger, production databases, you should increase the following shell limits to the following values recommended by Oracle:

```
nofile = 65536
```

To verify the above command execute **`ulimit -n`**.

```
nproc  = 16384
```

To verify the above command execute **`ulimit -u`**.

The **`nofile`** option denotes the maximum number of open file descriptors, and **`nproc`** denotes the maximum number of processes available to a single user.

To see all shell limits, execute:

```
ulimit -a
```

The following procedures and links show how to increase these parameters for the **`oracle`** user account:

- For more information on **`nofile`** and how to increase the limit, see *Section 11.1, “Limiting Maximum Number of Open File Descriptors for the Oracle User”*.
- For information on **`nproc`** and how to increase the limit, see *Section 11.2, “Limiting Maximum Number of Processes Available for the Oracle User”*.

Creating Oracle User Accounts

To create the oracle account and groups, execute the following commands:

```
su - root
groupadd dba          # group of users to be granted SYSDBA system privilege
groupadd oinstall     # group owner of Oracle files
useradd -c "Oracle software owner" -g oinstall -G dba oracle
passwd oracle
```

For more information on the "**oinstall**" group account, see *when to use "OINSTALL" group during install of oracle*[1].

[1] https://metalink.oracle.com/oracleinstall/oracle8i/genericunix.html#Uoui

Creating Oracle Directories

For Oracle 10g you only need to create the directory for **$ORACLE_BASE**:

```
su - root
mkdir -p /u01/app/oracle
chown oracle.oinstall /u01/app/oracle
```

If you want to comply with Oracle's Optimal Flexible Architecture (OFA), then you do not want to place the database files in the **/u01** directory but in another directory, file system or disk such as **/u02**:

```
su - root
mkdir -p /u02/oradata/orcl
chown oracle.oinstall /u02/oradata/orcl
```

In this example, "**orcl**" stands for the name of the database which will also be the name of the instance. This is typically the case for single instance databases.

24.1. Optimal Flexible Architecture (OFA) for 10g R1 (10.1.0.2)

The OFA standard is a guideline created by Oracle to ensure reliable Oracle installations. For Oracle 10g Database, the OFA recommended Oracle home path has changed.

The home path for the first 10g (10.1.0) database installation on a system would be:

```
/u01/app/oracle/product/10.1.0/db_1
```

If you would install a second Oracle 10g Database 10g (10.1.0) on the same system, the Oracle home directory should be as follows:

```
/u01/app/oracle/product/10.1.0/db_2
```

If the Oracle 10g software is not owned by the user **oracle** but by the user "**oraowner**", then the path of the Oracle home directory would be:

```
/u01/app/oraowner/product/10.1.0/db_1
/u01/app/oraowner/product/10.1.0/db_2
```

The standard directory name for Oracle 10g is "**app**":

```
/u01/app/oracle/product/10.1.0/db_1
```

Oracle recommends to use mount points such as **/u01**, **/u02**, and so on, complies with the OFA guidelines. But others can be used, for example:

```
/disk_1/app/oracle/product/10.1.0/db_1
```

The subtree for database files not stored in ASM disk groups should be named as follows:

```
/u02/oradata/<db_name_1>
/u02/oradata/<db_name_2>
```

```
/u03/oradata/<db_name_1>
/u03/oradata/<db_name_2>
```

The mount point **/u01** should be used for the Oracle software only. **/u02**, **/u03**, **/u04** and so on, should be used for the database files. **The db_name** represents the **DB_NAME** initialization parameter which is typically the same as the **SID** name for single instance databases.

Setting Oracle Environments

Since the Oracle Universal Installer (OUI) "runInstaller" is run from the oracle account, some environment variables must be configured for this account before OUI is started.

Execute the following commands for the **Bash** which is the default shell on Red Hat Enterprise Linux, to verify your shell run: **echo $SHELL**):

```
su - oracle
export ORACLE_BASE=/u01/app/oracle
export ORACLE_SID=orcl
```

Note

If **ORACLE_BASE** is used, then Oracle recommends that you do not set the **ORACLE_HOME** environment variable but that you choose the default path suggested by the OUI. You can set and use **ORACLE_HOME** after you finished running OUI.

The environment variables **ORACLE_HOME** and **TNS_ADMIN** should not be set. If you set these environment variables, you can unset them by running the following commands:

```
unset ORACLE_HOME
unset TNS_ADMIN
```

To have these environment variables set automatically each time you log on as **oracle**, you can add these environment variables to the **~oracle/.bash_profile** file which is the user start up file for the **Bash** on Red Hat Enterprise Linux. To do this you could simply copy and paste the following commands to make these settings permanent for your **oracle** accounts **Bash** shell:

```
su - oracle
cat >> ~oracle/.bash_profile << EOF
export ORACLE_BASE=/u01/app/oracle
export ORACLE_SID=orcl
EOF
```

Installing Oracle Database 10g

This Chapter will guide you through the process of installing an Oracle 10g Database.

26.1. Installing Oracle 10g on a Remote Linux Server

If you want to install Oracle on a remote server, then you need to relink **X** to your local desktop. The easiest way to do this is to use the "X11 forwarding" feature of **ssh**. This means that you do not have to run **xhost** and set the **DISPLAY** environment variable. Here is an example how to make use of the X11 forwarding feature of **ssh**. Simply run the following command from your local desktop:

```
$ ssh -X oracle@oracle_remote_server_name
```

Now when you run any GUI tool on the remote server, it will automatically be linked to your local desktop. If this is not working, verify that the **ForwardX11** setting is not set to "**no**" in **/etc/ssh/ssh_config** on the remote server:

```
su - root
# grep ForwardX11 /etc/ssh/ssh_config | grep -v "^#"
ForwardX11 yes
#
```

26.2. Installing Oracle 10gR2 Cluster Ready Services (CRS) with MPIO

There is a bug installing Oracle 10gR2 CRS services using MPIO(Multipath I/O). The bug has been reported to Oracle and the bug number and description states:

```
BUG 5005148 - CANNOT USE BLOCK DEVICES IN VOTING DISK AND OCR DURING 10G RAC INSTALLATION
```

This has been fixed in Oracle 11g.

Before attempting to install CRS, ensure that all firewalls are disabled:

```
service iptables stop
chkconfig iptables off
```

There are workarounds to successfully install CRS using MPIO. MPIO has to be disabled during installation. After the Vipca portion of the CRS installation is complete, the MPIO can be turned back on, restart CRS services. In the following examples using MPIO, friendly names are used for ease of manageability. In this example Oracle 10g R2 CRS is using internal redundancy for the **OCR** and **VOTING** disks.

Read **/usr/share/doc/device-mapper-multipath-0.4.5/Multipath-usage.txt** for more information on configuring multipath.

First, determine which block devices will be associated with MPIO devices with your OCR or Voting Disks and save the output to a log file.

```
# multipath -l > multipath.log
```

```
ocr1 (36006016054141500beac25dcc436dc11)
[size=1 GB][features="0"][hwhandler="1 emc"]
\_ round-robin 0 [active]
\_ 1:0:1:0 sdc 8:32  [active]
\_ 2:0:1:0 sdn 8:208 [active]

ocr2 (36006016054141500a898dae7c436dc11)
[size=2 GB][features="0"][hwhandler="1 emc"]
\_ round-robin 0 [active]
\_ 1:0:1:1 sdd 8:48  [active]
\_ 2:0:1:1 sdo 8:224 [active]

vote1 (360060160541415004 0a48bf2c436dc11)
[size=3 GB][features="0"][hwhandler="1 emc"]
\_ round-robin 0 [active]
\_ 1:0:1:2 sde 8:64  [active]
\_ 2:0:1:2 sdp 8:240 [active]

vote2 (36006016054141500c0fc0fffc436dc11)
[size=4 GB][features="0"][hwhandler="1 emc"]
\_ round-robin 0 [active]
\_ 1:0:1:4 sdg 8:96  [active]
\_ 2:0:1:4 sdr 65:16 [active]

vote3 (3600601605414150034 0b8309c536dc11)
[size=5 GB][features="0"][hwhandler="1 emc"]
\_ round-robin 0 [active]
\_ 1:0:1:5 sdh 8:112 [active]
\_ 2:0:1:5 sds 65:32 [active]
```

Delete raw links to Multipath Devices on all nodes:

```
rm -f  /dev/raw/*
```

Shut down multipath:

```
multipath -F  #shuts off Multipath
```

Looking at the multipath.log output file, you can see which block devices will used for Oracle CRS. To be sure you have the correct disk on all nodes, look at **`/proc/partitons`**. In this installation example, each device has a unique size so it is easier to map. Now link the block devices to the raw devices on all nodes by modifying **`/etc/sysconfig/rawdevices`**. Run **`ll`** on the device to get the major minor number for a block device, as seen below.

```
# ll /dev/sdc1
brw-rw----  1 root disk 8, 33 Oct  1 12:46 /dev/sdc1     #ocr1
# ll /dev/sdd1
brw-rw----  1 root disk 8, 49 Oct  1 12:47 /dev/sdd1     #ocr2
# ll /dev/sde1
brw-rw----  1 root disk 8, 65 Oct  1 12:47 /dev/sde1  #vote1
# ll /dev/sdg1
brw-rw----  1 root disk 8, 97 Oct  1 12:48 /dev/sdg1  #vote2
# ll /dev/sdh1
brw-rw----  1 root disk 8, 113 Oct  1 12:49 /dev/sdh1  #vote3
```

Change the configuration file **`/etc/sysconfig/rawdevices`** using vi or your favorite editor and add the following block devices.

```
$ vi /etc/sysconfig/rawdevices
```

```
/dev/raw/raw1   8 33
/dev/raw/raw2   8 49
/dev/raw/raw3   8 65
/dev/raw/raw4   8 97
/dev/raw/raw5   8 113
```

On the other nodes run **blockdev** so the partition table are read again and created:

```
# blockdev --rereadpt /dev/sdc
# blockdev --rereadpt /dev/sdd
# blockdev --rereadpt /dev/sde
# blockdev --rereadpt /dev/sdg
# blockdev --rereadpt /dev/sdh
```

Double check on the other nodes that the minor numbers for the block devices match. The second number in the example below, 8 in this case, is the minor number and 1 is the major number.

```
$ ll /dev/sdh1
brw-rw----  1 root disk 8, 113 Oct  1 13:14 /dev/sdh1
```

Copy **/etc/sysconfig/rawdevices** to other nodes then restart it. You will see the created raw devices in **/dev/raw**.

```
# /etc/rc5.d/S56rawdevices start
Assigning devices:
/dev/raw/raw1  -->   8 33
/dev/raw/raw1:  bound to major 8, minor 33
/dev/raw/raw2  -->   8 49
/dev/raw/raw2:  bound to major 8, minor 49
/dev/raw/raw3  -->   8 65
/dev/raw/raw3:  bound to major 8, minor 65
/dev/raw/raw4  -->   8 97
/dev/raw/raw4:  bound to major 8, minor 97
/dev/raw/raw5  -->   8 113
/dev/raw/raw5:  bound to major 8, minor 113
done
```

Now check that the permissions are set for the Oracle installation.

```
# ll /dev/raw
total 0
crwxrwxrwx  1 oracle dba 162, 1 Oct  1 13:13 raw1
crwxrwxrwx  1 oracle dba 162, 2 Oct  1 13:13 raw2
crwxrwxrwx  1 oracle dba 162, 3 Oct  1 13:13 raw3
crwxrwxrwx  1 oracle dba 162, 4 Oct  1 13:13 raw4
crwxrwxrwx  1 oracle dba 162, 5 Oct  1 13:13 raw5
```

Now you can commence an Oracle Clusterware installation.

Note

If during the installation process, Oracle says the raw devices are busy, then Multipath was left on in one of the nodes.

After the **VIPCA** portion of the Oracle CRS is complete, you need to comment out the raw devices you created and copy **/etc/sysconfig/rawdevices** to the other nodes:

```
#/dev/raw/raw1  8 33
#/dev/raw/raw2  8 49
#/dev/raw/raw3  8 65
#/dev/raw/raw4  8 97
#/dev/raw/raw5  8 113
```

Check to be sure the Oracle Notification Services has started on all nodes.

```
# ps -ef |grep ons
oracle   20058     1  0 13:49 ?        00:00:00 /ora/crs/opmn/bin/ons -d
oracle   20059 20058  0 13:49 ?        00:00:00 /ora/crs/opmn/bin/ons -d
root     20812 28286  0 13:50 pts/2    00:00:00 grep ons
```

The installation is now complete. Now you can to turn off Oracle CRS and start using MPIO.

Note

It is best to do this process one node at a time. No need for reboots during this process.

Shutdown the Oracle CRS Process.

```
# /ora/crs/bin/crsctl stop crs
Stopping resources.
Successfully stopped CRS resources
Stopping CSSD.
Shutting down CSS daemon.
Shutdown request successfully issued.
```

Delete the raw devices created for Oracle CRS installation.

```
$ rm -f /dev/raw/*
```

Turn on Multipath I/O (MPIO) again.

```
$ multipath
```

Create a simple script in **`/etc/rc5.d/`** (before CRS Starts but after MPIO starts) to bind raw devices using MPIO.

```
num="1"
for i in `ls /dev/mpath/ocr?p? | sort`
do
raw /dev/raw/raw${num} $i
let "num = $num + 1"
done
num="3"
for i in `ls /dev/mpath/vote?p? | sort`
do
raw /dev/raw/raw${num} $i
let "num = $num + 1"
done
```

Then run the script, **`/etc/rc5.d/`**, that you just created.

```
# /etc/rc5.d/S57local start
```

```
/dev/raw/raw1:  bound to major 253, minor 25
/dev/raw/raw2:  bound to major 253, minor 21
/dev/raw/raw3:  bound to major 253, minor 22
/dev/raw/raw4:  bound to major 253, minor 23
/dev/raw/raw5:  bound to major 253, minor 24
```

Verify the correct permissions are all set.

```
# ll /dev/raw
total 0
crwxrwxrwx  1 oracle dba 162, 1 Oct  1 13:58 raw1
crwxrwxrwx  1 oracle dba 162, 2 Oct  1 13:58 raw2
crwxrwxrwx  1 oracle dba 162, 3 Oct  1 13:58 raw3
crwxrwxrwx  1 oracle dba 162, 4 Oct  1 13:58 raw4
crwxrwxrwx  1 oracle dba 162, 5 Oct  1 13:58 raw5
```

Restart CRS and be sure the ONS processes all start. This ensures Oracle CRS is functional on this node

```
# /ora/crs/bin/crsctl start crs
Attempting to start CRS stack
The CRS stack will be started shortly
```

Repeat for the process for the rest of the nodes. Now the Oracle CRS installation is complete using Multipath I/O.

Once complete, you can check the status of the nodes

```
# crs_stat -t
Name           Type                 Target         State          Host
------------------------------------------------------------------------
ora....t08.gsd application         ONLINE        ONLINE     et-virt08
ora....t08.ons application         ONLINE        ONLINE     et-virt08
ora....t08.vip application         ONLINE        ONLINE     et-virt08
ora....t09.gsd application         ONLINE        ONLINE     et-virt09
ora....t09.ons application         ONLINE        ONLINE     et-virt09
ora....t09.vip application         ONLINE        ONLINE     et-virt09
ora....t10.gsd application         ONLINE        ONLINE     et-virt10
ora....t10.ons application         ONLINE        ONLINE     et-virt10
ora....t10.vip application         ONLINE        ONLINE     et-virt10
ora....t11.gsd application         ONLINE        ONLINE     et-virt11
ora....t11.ons application         ONLINE        ONLINE     et-virt11
ora....t11.vip application         ONLINE        ONLINE     et-virt11
#
```

26.3. Starting Oracle Universal Installer

Insert the Oracle CD that contains the image of the downloaded file **`ship.db.lnx32.cpio`**, or change to the directory that contains the image directory Disk1.

Before you execute **`runInstaller`**, make sure the Oracle environment variables are set, see Setting Oracle Environments. You can verify the settings by running the set command:

```
su - oracle
oracle$ set
```

If you install Oracle 10g from a CD, mount the CD by running the following commands in another terminal: On Red Hat Enterprise Linux 2.1 execute:

```
$ su - root
# mount /mnt/cdrom
```

On Red Hat Enterprise Linux 3 and 4 execute:

```
$ su - root
# mount /media/cdrom
```

To execute **runInstaller** from the mounted CD, run the command corresponding to your version as the **oracle** user.

On Red Hat Enterprise Linux 2.1 execute:

```
oracle$ /mnt/cdrom/runInstaller
```

On Red Hat Enterprise Linux 3 and 4 execute:

```
oracle$ /media/cdrom/runInstaller
```

26.4. Using Oracle Universal Installer (OUI)

The following example shows how to install x86 Oracle 10g Release 1 Database Software and a "General Purpose" database. *Please note, the screens and questions will look different if you install 10g R2 or 64 bit 10g R1 database.* - Welcome Screen:

```
- Basic Installation:           Check for the default
- Oracle Home Location:         Use default:
                                /u01/app/oracle/product/10.1.0/db_1
- Installation Type:            Enterprise Edition
- UNIX DBA Group:               Use default: dba
- Create Starter Databases:     Check it which is the default
    - Global Database Name:     orcl
- Database password:            Type in the password for SYS, SYSTEM,
                                SYSMAN, and DBSNMP accounts
- Advanced Installation:        Not checked it this example
                                Click Next
```

- Specify Inventory directory and credentials:

```
- Full path of the inventory directory:
          Use the default:     /u01/app/oracle/oraInventory
- Specify Operating System group name:
          Use default:         oinstall
```

Click **Next**

- A window pops up to run the **orainstRoot.sh** script:

Run the script in another terminal:

```
su - root
# /u01/app/oracle/oraInventory/orainstRoot.sh
Creating the Oracle inventory pointer file (/etc/oraInst.loc)
Changing groupname of /u01/app/oracle/oraInventory to oinstall.
#
```

Then click **Continue**

- Product-specific Prerequisite Checks:

```
Verify that all checks have been passed.
```

Make sure that the status of each Check is set to "Succeeded".

On Red Hat Enterprise Linux AS 4 ignore the warnings for binutils, gcc, and openmotif and proceed.

If a check fails, see *Chapter 29, Oracle 10g and Linux Installation Errors and Troubleshooting*.

Note

The "**Retry**" button does not work after you have fixed one of the failed checks.

Click **Next**

- Select Database Configuration then select "General Purpose".

Click **Next**

- Specify Database Configuration Options:

```
- Global Database Name: for example use "orcl".
- SID: for example use "orcl".
```

Click **Next**

- Select Database Management Option then select "Use Database Control for Database Management".

Click **Next**

- Specify Database File Storage Option

```
   Select "File System" in this example.
       - File System
           - Specify Database file location: /u01/app/oracle/oradata/
               If you want to comply with OFA, you might want to select
 another mount point than '/u01', for example /u02/oradata.
```

Click **Next**

- Specify Backup and Recovery Options then select "Do no enable Automated Backups" for this example.

Click **Next**

- Specify Database Schema Passwords

Make sure that the passwords do not start with a number. this will cause an error message "ORA-00988 missing or invalid password" or something similar later.

Click **Next**

- Summary

Click **Install**

If **Enterprise Manager** configuration fails due to port allocation problems, read *Chapter 29, Oracle 10g and Linux Installation Errors and Troubleshooting*.

When a window pops up to run the **root.sh** script, execute the script in another terminal as root:

```
su - root
# /u01/app/oracle/product/10.1.0/db_1/root.sh
Running Oracle10 root.sh script...

The following environment variables are set as:
 ORACLE_OWNER= oracle
 ORACLE_HOME=  /u01/app/oracle/product/10.1.0/db_1

Enter the full path name of the local bin directory:
[/usr/local/bin]:
 Copying dbhome to /usr/local/bin ...
 Copying oraenv to /usr/local/bin ...
 Copying coraenv to /usr/local/bin ...
Creating /etc/oratab file...
Adding entry to /etc/oratab file...
Entries will be added to the /etc/oratab file as needed by
Database Configuration Assistant when a database is created.
Finished running generic part of root.sh script.
Now product-specific root actions will be performed.
/var/opt/oracle does not exist. Creating it now.
/etc/oracle does not exist. Creating it now.
Successfully accumulated necessary OCR keys.
Creating OCR keys for user 'root', privgrp 'root'..
Operation successful.
Oracle Cluster Registry for cluster has been initialized

Adding to inittab
Checking the status of Oracle init process...
Expecting the CRS daemons to be up within 600 seconds.
CSS is active on these nodes.
 mars
CSS is active on all nodes.
Oracle CSS service is installed and running under init(1M)
#
```

Click **OK**

You have now reached the end of the Installation, click the **Exit** button.

26.5. Updating after the Oracle Universal Installer

After Oracle 10g has been installed, make sure that **ORACLE_HOME**, **PATH**, and **LD_LIBRARY_PATH** are set for the **oracle** account.

Note

The path for **ORACLE_HOME** might be different on your system! Also note that **LD_LIBRARY_PATH** is needed for some Oracle binaries such as **sysresv**.

For 10g R1 (10.1.0.3) you may want to add the following lines to the **~oracle/.bash_profile** file:

```
export ORACLE_HOME=$ORACLE_BASE/product/10.1.0/db_1
export PATH=$PATH:$ORACLE_HOME/bin
export LD_LIBRARY_PATH=$ORACLE_HOME/lib
```

For 10g R2 (10.2.0.1.0) you may want to add the following lines to the **~oracle/.bash_profile** file:

```
export ORACLE_HOME=$ORACLE_BASE/oracle/product/10.2.0/db_1
export PATH=$PATH:$ORACLE_HOME/bin
export LD_LIBRARY_PATH=$ORACLE_HOME/lib
```

After that run the following command to set all environment variables in **~oracle/.bash_profile**:

```
$ . ~oracle/.bash_profile
```

This command will add the environment variables to the **~oracle/.profile** and source in the file for the current shell by executing "**. ~oracle/.bash_profile**".

Note

Do not add a trailing "**/**" on the **ORACLE_HOME** environment variable. Otherwise you will get the error "**ORACLE not available**" when you try to connect to sys, see *Chapter 29, Oracle 10g and Linux Installation Errors and Troubleshooting*.

Oracle Post Installation Tasks

Important

Before you continue, make sure you followed the steps at *Section 26.5, "Updating after the Oracle Universal Installer"*.

27.1. Startup and Shutdown of the Oracle 10g Database

This section contains details on how to start up and shut down your Oracle Database. Note that the

Start the database:

```
oracle$ sqlplus /nolog
SQL> connect / as sysdba
SQL> startup
```

Shut down the database:

```
oracle$ sqlplus /nolog
SQL> connect / as sysdba
SQL> shutdown
```

The slash connects you to the schema owned by **SYS**. In the above example you will be connected to the schema owned by **SYS** with the privilege **SYSDBA**. **SYSDBA** gives you the following privileges:

- sysoper privileges WITH ADMIN OPTION.
- create database.
- recover database until.

27.2. Shutdown of other Oracle 10g Background Processes

If you installed a pre-configured database using OUI, then several Oracle background processes are now running on your server. Execute the following command to see the background processes:

```
ps -ef
```

To shutdown the Oracle background processes after an Oracle Database 10g installation, you can execute the following commands:

- iSQL*Plus

  To stop iSQL*Plus, run:

  ```
  su - oracle
  isqlplusctl stop
  ```

- Database Management Processes

  During the installation of Oracle 10g, OUI offered two Database Management Options:

If you selected "Database Control for Database Management", then the Oracle Enterprise Manager Database Control (Database Control) can be shutdown with the following command which stops both the agent and the Oracle Containers for Java (OC4J) management service:

```
su - oracle
emctl stop dbconsole
```

If you selected "Grid Control for Database Management" which is used for full "Grid Control" installations, then the Oracle Management Agent (standalone agent) for the Oracle Enterprise Manager Grid Control (Grid Control) can be stopped with the following command:

```
su - oracle
emctl stop agent
```

- Oracle Net Listener

  To stop the listener, run:

  ```
  su - oracle
  lsnrctl stop
  ```

- Cluster Synchronization Services (CSS)

  To shutdown Oracle Cluster Synchronization Services (CSS) daemon, run:

  ```
  su - root
  /etc/rc.d/init.d/init.cssd stop
  ```

Tips and Hints for Oracle 10g on Red Hat Enterprise Linux

To reinstall Oracle 10g after a failed installation attempt, you might want to execute the following commands. Make sure you first used the "De-installation" option in the OUI.

```
su - root
export ORACLE_HOME=/u01/app/oracle/product/10.1.0/db_1
. $ORACLE_HOME/bin/localconfig delete
```

The above line stops the Oracle CSS daemon and deletes the configuration.

```
rm -rf /u01/app/oracle/*

rm -f /etc/oraInst.loc /etc/oratab
rm -rf /etc/oracle
rm -f /etc/inittab.cssd
rm -f /usr/local/bin/coraenv /usr/local/bin/dbhome /usr/local/bin/oraenv
```

Make sure you comment out **`ORACLE_HOME`** from **`~oracle/.bash_profile`**.

Oracle 10g and Linux Installation Errors and Troubleshooting

This chapter presents many common Oracle 10g problems and other issues.

Note

Most of the issues are caused by not following the installation procedure correctly. Some errors are caused by not using an Oracle supported Linux OS version such as Red Hat Enterprise Linux.

The Installation log file can be found in **`$ORACLE_BASE/oraInventory/logs`**. The Database Creation log file can be found in **`$ORACLE_BASE/admin/$ORACLE_SID/create`**.

An Error While Starting the Oracle Universal Installer

You may receive the following error while running the **Oracle Universal Installer:**

```
Starting Oracle Universal Installer...
Checking installer requirements...

Checking operating system version: must be redhat-2.1, UnitedLinux-1.0 or
    redhat-3
    Failed <<<<

Exiting Oracle Universal Installer, log for this session can be found at ...
```

A solution for how to fix this can be found at *Section 20.8, "Verifying and Updating the redhat-release File"*.

Note for All Errors

The "**Retry**" in the "**Product-specific Prerequisite Checks**" window does not work. So you either set it manually to Passed or you restart **OUI**.

Checking for gcc-2.96; found Not found. Failed >>>>

Refer to *Section 19.3, "Verifying Software Packages (RPMs)"* for information on this problem and how to solve it.

Checking for shmmax=2147483648; found shmmax=33554432. Failed >>>>

Solution: Increase the **`shmmax`** kernel parameter.

See *Section 19.4, "Verifying Kernel Parameters"* for information on this problem and how to solve it.

Checking for semopm=100; found semopm=32. Failed <<<<

Increase the **`semopm`** kernel parameter.

See *Section 19.4, "Verifying Kernel Parameters"* for information on this problem and how to solve it.

Checking for filemax=65536; found filemax=26163. Failed <<<<

Increase the **file-max** kernel parameter.

See *Section 19.4, "Verifying Kernel Parameters"* for information on this problem and how to solve it.

ORA-01034: ORACLE not available, ORA-27101: shared memory realm does not exist, Linux Error: 2: No such file or directory, or ORA-01034: ORACLE not available

First check if **ORACLE_SID** is set correctly. If the **ORACLE_SID** is set correctly, then you probably have a trailing slash "/" on the **ORACLE_HOME** environment variable. Remove it and try again to connect to **sys**. For example, change **ORACLE_HOME=/u01/app/oracle/product/10.1.0/db_1/** to **ORACLE_HOME=/u01/app/oracle/product/10.1.0/db_1**.

ORA-00988 missing or invalid password(s)

During the Oracle 10g installation you probably provided a password for the Oracle database accounts that started with a number. Ignore this error message and change the password when you are done with the Oracle 10g installation.

sysresv -i outputs sysresv: error while loading shared libraries: libclntsh.so.10.1: cannot open shared object file: No such file or directory

Make sure LD_LIBRARY_PATH is set to $ORACLE_HOME/lib by executing:

```
oracle$ export LD_LIBRARY_PATH=$ORACLE_HOME/lib
```

X11 connection rejected because of wrong authentication X connection to localhost:10.0 broken (explicit kill or server shutdown)

To rectify this problem, try to log in to the remote Oracle server again by using the "X11 forward" feature of ssh. Execute the following command, after changing oracle_remote_server_name to your server, from your local desktop:

```
$ ssh -X oracle@oracle_remote_server_name
```

Now when you try to run any GUI tool on the remote server, it should automatically be relinked to your local desktop. If this is not working, verify that the **ForwardX11** setting is not set to "**no**" in **/etc/ssh/ssh_config** on your remote server:

```
su - root
# grep ForwardX11 /etc/ssh/ssh_config | grep -v "^#"
  ForwardX11 yes
#
```

Note

If you use newer Red Hat Enterprise Linux versions as your desktop and you want to install the database on another machine, then you need to set the **DisallowTCP** entry in **/etc/X11/gdm/gdm.conf** for the GNOME Display Manager to read:

```
DisallowTCP=false
```

After that you need to restart your X server. You can do this with the **init** command:

su - root init 3 init 5

If you are using **telnet** you will have to set the **DISPLAY** variable manually.

Recovery Manager(rman) hangs

You are probably running the wrong **rman** binary which belongs to the **XFree86-devel** RPM:

```
$ which rman
/usr/X11R6/bin/rman
```

ORA-00988 missing or invalid password(s)

During the Oracle 10g installation you probably provided a password for the Oracle database accounts that started with a digit number. Ignore this error message and change the password when you are done with the Oracle 10g installation.

./runInstaller Crashes

The Oracle installer ./run Installer crashes with the

```
Exception java.lang.UnsatisfiedLinkError: /tmp/OraInstall2005-06-15_07-36-25AM/jre/1.4.2/lib/
i386/libawt.so:
libXp.so.6: cannot open shared object file: No such file or directory occurred..
java.lang.UnsatisfiedLinkError: /tmp/OraInstall2005-06-15_07-36-25AM/jre/1.4.2/lib/i386/
libawt.so:
libXp.so.6: cannot open shared object file: No such file or directory
 at java.lang.ClassLoader$NativeLibrary.load(Native Method)
 at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1560)
 at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1477)
```

You may receive this error message on Red Hat Enterprise Linux 3 x86-64, Red Hat Enterprise Linux 4 x86-64, and on other systems. Even though you most probably have **/usr/X11R6/lib64/libXp.so.6** installed on your system, this error messages is complaining that it can not find the **libXp.so.6** shared library for i386:

```
/tmp/OraInstall2005-06-15_07-36-25AM/jre/1.4.2/lib/i386/libawt.so: libXp.so.6: cannot open
 shared object file: No such file or directory
```

Follow the instructions below to install the i386 XFree86-libs package (XFree86-libs-4.3.0-81.EL.i386.rpm or newer) on Red Hat Enterprise Linux 3 x86-64 with 10g (10.1.0.3) . In order to satisfy dependencies for this i386 package, you may need to install a few other i386 RPMs as well:

```
# rpm -ivh XFree86-libs-4.3.0-81.EL.i386.rpm \
    XFree86-Mesa-libGL-4.3.0-81.EL.i386.rpm \
    expat-1.95.5-6.i386.rpm \
    fontconfig-2.2.1-13.i386.rpm \
    freetype-2.1.4-4.0.i386.rpm \
    zlib-1.1.4-8.1.i386.rpm
```

In order to install 10g (10.1.0.3) on Red Hat Enterprise Linux 4 x86-64 update 1 you must install the i386 **xorg-x11-deprecated-libs** package (**xorg-x11-deprecated-**

libs-6.8.2-1.EL.13.6.i386.rpm). The installation depends on the following packages, to install these packages execute the follow commands:

```
# rpm -Uvh xorg-x11-deprecated-libs-6.8.2-1.EL.13.6.i386.rpm \
    xorg-x11-libs-6.8.2-1.EL.13.6.i386.rpm \
    xorg-x11-Mesa-libGL-6.8.2-1.EL.13.6.i386.rpm \
    expat-1.95.7-4.i386.rpm \
    fontconfig-2.2.3-7.i386.rpm \
    freetype-2.1.9-1.i386.rpm \
    zlib-1.2.1.2-1.i386.rpm
```

Installing 10g R2 (10.2.0.1.0) on Red Hat Enterprise Linux 4 x86-64 Update 2 requires the i386 **xorg-x11-deprecated-libs** package (**xorg-x11-deprecated-libs-6.8.2-1.EL.13.20.i386.rpm**). In order to satisfy dependencies for this i386 package, you may have to install a few other i386 RPMs as well:

```
rpm -Uvh xorg-x11-deprecated-libs-6.8.2-1.EL.13.20.i386.rpm \
    xorg-x11-libs-6.8.2-1.EL.13.20.i386.rpm \
    xorg-x11-Mesa-libGL-6.8.2-1.EL.13.20.i386.rpm \
    expat-1.95.7-4.i386.rpm \
    fontconfig-2.2.3-7.i386.rpm \
    freetype-2.1.9-1.i386.rpm \
    zlib-1.2.1.2-1.2.i386.rpm
```

An Oracle 10g R2 (10.2.0.1.0) install on Red Hat Enterprise Linux 4 x86 Update 2 system requires the following RPM:

```
rpm -Uvh xorg-x11-deprecated-libs-6.8.2-1.EL.13.20.i386.rpm
```

After you have installed the appropriate RPMs, restart the installation by running **.runInstaller** again.

An error in relinking during the Oracle install

You may get this error message or a similar one when installing 64 bit 10g on Red Hat Enterprise Linux4 x86-64.

```
make -f /u01/app/oracle/OraHome_1/sysman/lib/ins_sysman.mk relink_sharedobj
 SHAREDOBJ=libnmemso
make[1]: Entering directory `/u01/app/oracle/OraHome_1/sysman/lib'
gcc -o /u01/app/oracle/OraHome_1/sysman/lib/libnmemso.so -m32 ...
...
/usr/bin/ld: crti.o: No such file: No such file or directory
collect2: ld returned 1 exit status
make[1]: *** [/u01/app/oracle/OraHome_1/sysman/lib/libnmemso.so] Error 1
```

On Red Hat Enterprise Linux4 U1 x86-64 to install 10g (10.1.0.3) you need the following i386 RPM to fix this problem:

```
# rpm -Uvh glibc-devel-2.3.4-2.9.i386.rpm
```

On Red Hat Enterprise Linux 4 U2 x86-64 to install 10g R2 (10.2.0.1.0) you need the following i386 RPM to fix this problem:

```
# rpm -Uvh glibc-devel-2.3.4-2.13.i386.rpm
```

Error while loading shared libraries: libaio.so.1: cannot open shared object file: No such file or directory

Make sure the **libaio** RPM is installed.

On Red Hat Enterprise Linux 3 x86:

```
# rpm -Uvh libaio-0.3.96-5.i386.rpm
```

On Red Hat Enterprise Linux 4 Update 2 x86-64:

```
# rpm -Uvh libaio-0.3.103-3.x86_64.rpm
```

Your version of **libaio** may be different.

Error in invoking target 'all_no_orcl'

The error text in full:

```
Error in invoking target 'all_no_orcl' of makefile '/u01/app/oracle/oracle/product/10.2.0/
db_1/rdbms/lib/ins_rdbms.mk'.
See '/u01/app/oracle/oraInventory/logs/installActions2005-11-13_01-07-04AM.log' for details.
```

The log file shows the following error:

```
INFO: gcc:
INFO: /usr/lib64/libstdc++.so.5: No such file or directory
INFO:

INFO: /u01/app/oracle/oracle/product/10.2.0/db_1/bin/genorasdksh:
 Failed to link liborasdkbase.so.10.2

INFO: make: *** [liborasdkbase] Error 1
```

On Red Hat Enterprise Linux4 U2 x86-64 with 10g R2 (10.2.0.1.0) install the following x86-64 RPM to fix this problem:

```
# rpm -Uvh compat-libstdc++-33-3.2.3-47.3.x86_64.rpm
```

Note

You may already have the "i386" **compat-libstdc++-33** RPM installed on your systems but you need the "x86-64" RPM to fix this problem. To verify which **compat-libstdc++-33** RPM you have installed on your system, run:

```
# rpm -q --qf '%{NAME}-%{VERSION}-%{RELEASE}  (%{ARCH})\n' compat-libstdc++-33
```

Error in invoking target 'all_no_orcl ihsodbc'

The full error message:

```
Error in invoking target 'all_no_orcl ihsodbc' of makefile '/u01/app/oracle/oracle/
product/10.2.0/db_1/rdbms/lib/ins_rdbms.mk'.
See '/u01/app/oracle/oraInventory/logs/installActions2005-07-24_09-25-22AM.log' for details.
```

The log file shows the following error:

```
INFO: Creating /u01/app/oracle/oracle/product/10.2.0/db_1/lib/liborasdkbase.so.10.2

INFO: gcc:
INFO: /usr/lib/libstdc++.so.5: No such file or directory
INFO:

INFO: /u01/app/oracle/oracle/product/10.2.0/db_1/bin/genorasdksh:
 Failed to link liborasdkbase.so.10.2
```

This means that the "33" version of the **`compat-libstdc++`** RPM is missing.

Installing 10g R2 (10.2.0.1.0) on Red Hat Enterprise Linux 4 Update 2 x86 you require the following RPM to fix this problem:

```
# rpm -Uvh compat-libstdc++-33-3.2.3-47.3.i386.rpm
```

You need the "33" version of the compat-libstdc++ RPM. For i386 there is also a "296" version of the **`compat-libstdc++`** RPM. Here are the two **`compat-libstdc++`** RPMs that come with Red Hat Enterprise Linux 4 Update 2:

```
compat-libstdc++-296-2.96-132.7.2.i386.rpm
compat-libstdc++-33-3.2.3-47.3.i386.rpm
```

After that hit **Retry** in the error dialog window.

Error in invoking target 'all_no_orcl' of makefile

The full error is listed below.

```
Error in invoking target 'all_no_orcl' of makefile '/u01/app/oracle/oracle/product/10.2.0/
db_1/rdbms/lib/ins_rdbms.mk'.
See '/u01/app/oracle/oraInventory/logs/installActions2005-11-13_01-25-49AM.log' for details.
```

The log file shows the following error:

```
INFO: /usr/bin/ld: /u01/app/oracle/oracle/product/10.2.0/db_1/lib/libirc.a(fast_memcpy.o):
relocation R_X86_64_PC32 against `_memcpy_mem_ops_method' can not be used
when making a shared object; recompile with -fPIC /usr/bin/ld: final link failed: Bad value
collect2: ld returned 1 exit status

INFO: /u01/app/oracle/oracle/product/10.2.0/db_1/bin/genorasdksh:
 Failed to link liborasdkbase.so.10.2
```

This error comes up when installing 10g R2 (10.2.0.1.0) on Red Hat Enterprise Linux 4 x86-64. Make sure to upgrade to Red Hat Enterprise Linux 4 Update 3, or higher, otherwise to download the **`binutils`** RPM from *https://rhn.redhat.com/* or from *http://oss.oracle.com/projects/compat-oracle/files/RedHat/*. After you have downloaded the RPM install it:

```
# rpm -Uvh --force binutils-2.15.92.0.2-13.0.0.0.2.x86_64.rpm
```

For more information on this bug, see *Bugzilla Bug 679*[1].

[1] http://sources.redhat.com/bugzilla/show_bug.cgi?id=679

ORA-12547: TNS:lost contact

There can be many reasons for this error. This can happen during the ASM instance start up when the **`libaio`** RPM is not installed on the system.

You can test this by running **`lsnrctl`** as the **`oracle`** user.

```
$ lsnrctl start
...

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC)))
   TNS-12547: TNS:lost contact
      TNS-12560: TNS:protocol adapter error
         TNS-00517: Lost contact
     Linux Error: 104: Connection reset by peeras
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=centauri)(PORT=1521)))
   TNS-12547: TNS:lost contact
      TNS-12560: TNS:protocol adapter error
         TNS-00517: Lost contact
     Linux Error: 104: Connection reset by peer
```

Make sure the `loopback` entry in **`/etc/hosts`** is not missing when you start the listener:

```
127.0.0.1       localhost.localdomain   localhost
```

Now try to run **`lsnrctl`** as the user **`oracle`** again.

Part III. Installing the Oracle9i 32 bit Database on Red Hat Enterprise Linux

Introduction and Information on Supported Setups for Oracle 9i

This article provides a step by step guide for installing Oracle9i databases on Red Hat Enterprise Linux. This guide covers the following Oracle Database and Red Hat Linux versions:

Oracle Database Version	Red Hat Enterprise Linux Version	Architecture
Oracle9i R2 (9.2.0.6.0)	4	x86 (32 bit)
Oracle9i R2 (9.2.0.4.0)	3	x86 (32 bit)
Oracle9i R2 (9.2.0.1.0)	2.1	x86 (32 bit)

Table 4. Table displaying Certified Red Hat Enterprise Linux For Oracle 10g

For Validation and Certification information, see *Oracle's Certification Matrices*[2].

[2] https://metalink.oracle.com/metalink/certify/certify.welcome

Preparing Red Hat Enterprise Linux for an Oracle Database Installation

This chapter will guide you though the pre-installation procedures for installing Oracle 9i on Red Hat Enterprise Linux

30.1. Unpacking and Downloading the Oracle9i Installation Files

Download Oracle9i for Linux from *http://otn.oracle.com/software/products/oracle9i/htdocs/linuxsoft.html*.

Here are two methods for unpacking the Oracle installation files, choose one. First method - this method uses less disk space and should be faster:

```
zcat lnx_920_disk1.cpio.gz | cpio -idmv
zcat lnx_920_disk2.cpio.gz | cpio -idmv
zcat lnx_920_disk3.cpio.gz | cpio -idmv
```

Second method - this method first uncompresses then unpacks the installation files:

```
gunzip lnx_920_disk1.cpio.gz lnx_920_disk2.cpio.gz lnx_920_disk3.cpio.gz Linux9i_Disk3.cpio.gz

cpio -idmv < lnx_920_disk1.cpio
cpio -idmv < lnx_920_disk2.cpio
cpio -idmv < lnx_920_disk3.cpio
```

You should now have 3 directories containing installation files, if **ls** looks like this it worked correctly:

```
$ ls
Disk1
Disk2
Disk3
```

30.2. Setting Swap Space

According to *Oracle9i Installation Guide Release 2*[1] a minimum of 512MB of RAM is required to install an Oracle9i Server. The swap space should be equal to RAM, or 1GB, whichever is greater. For more information on correctly sizing the swap space for your database, read *Chapter 6, Swap Space*..

To check the memory, run:

```
grep MemTotal /proc/meminfo
```

To check the swap space, run:

```
cat /proc/swaps
```

[1] http://download-east.oracle.com/docs/html/A96167_01/pre.htm#sthref104

30.3. Setting Shared Memory

For Oracle9i installations, the maximum shared memory size must be increased. If it is too small, the Oracle Database **Configuration Assistant** will display the following error message:

```
ORA-27123: unable to attach to shared memory segment.
```

To increase the **shmmax** setting, execute the following command:

```
$ su - root
# cat /proc/sys/kernel/shmmax
33554432
# echo `expr 1024 \* 1024 \* 1024` > /proc/sys/kernel/shmmax
# cat /proc/sys/kernel/shmmax
1073741824
```

It is recommended to increase the **shmmax** setting permanently for Oracle. For more information on optimizing shared memory settings for Oracle databases on Linux, see *Chapter 7, Setting Shared Memory*. These parameters apply to all Red Hat Enterprise Linux versions.

30.4. Examining Temporary(/tmp) Space

The Oracle Universal Installer requires up to 400 MB of free space in the /tmp directory.

To find out the space in /tmp, run:

```
$ df /tmp
```

If you do not have enough space in the **/tmp** directory, you can temporarily create a **tmp** directory in another file system. Here are the steps for doing it:

```
su - root
mkdir /<AnotherFilesystem>/tmp
chown root.root /<AnotherFilesystem>/tmp
chmod 1777 /<AnotherFilesystem>/tmp
export TEMP=/<AnotherFilesystem>
export TMPDIR=/<AnotherFilesystem>
```

The **TEMP=/<AnotherFilesystem>** file is used by Oracle. The TMPDIR=/<AnotherFilesystem> file is used by Linux programs like the linker "**ld**".

When you are done with your Oracle installation, shut down Oracle and remove the temporary directory:

```
su - root
rmdir /<AnotherFilesystem>/tmp
unset TEMP
unset TMPDIR
```

30.5. Sizing Oracle Disk Space

You will need about 2.5 GB for the database software. If you perform a typical database installation and not a customized database installation, then you will need about 3.5 GB of disk space.

Verifying Required Packages(RPMs)

You will need certain RPM development packages for the Oracle installer to build Oracle modules, otherwise you will get error messages similar to this one:

```
Error in invoking target ntcontab.o of makefile
/u01/app/oracle/product/9.2.0/network/lib/ins_net_client.mk
```

Note

Always ensure you use the latest stable versions of RPM.

See the Oracle9i Release Notes Release 2 (9.2.0.4.0) for Linux x86 for the list of required RPMs.

31.1. Required Packages for Red Hat Advanced Server 2.1

Ensure the following development packages are installed:

```
rpm -q gcc cpp compat-libstdc++ glibc-devel kernel-headers binutils
```

Most of these packages will be missing on Red Hat Advanced Server 2.1 if the "Software Development" package was not selected during the OS install. If these RPMs are missing, execute the command below, remember your versions may be newer. If they are not newer than those seen below update to a more recent version.

```
rpm -ivh cpp-2.96-108.1.i386.rpm \
    glibc-devel-2.2.4-26.i386.rpm      \
    kernel-headers-2.4.9-e.3.i386.rpm  \
    gcc-2.96-108.1.i386.rpm            \
    binutils-2.11.90.0.8-12.i386.rpm
```

31.2. Required Packages for Red Hat Enterprise Linux 3

Ensure the following packages are installed by executing the command below:

```
rpm -q make            \
    binutils               \
    gcc                    \
    cpp                    \
    glibc-devel            \
    glibc-headers          \
    glibc-kernheaders      \
    compat-db              \
    compat-gcc             \
    compat-gcc-c++         \
    compat-libstdc++       \
    compat-libstdc++-devel \
    gnome-libs             \
    openmotif21            \
    setarch
```

31.3. Required Packages for Red Hat Enterprise Linux 4

Ensure the following packages are installed by executing the following command:

```
rpm -q make                    \
    compat-db                      \
    compat-gcc-32                  \
    compat-gcc-32-c++              \
    compat-oracle-rhel4            \
    compat-libcwait                \
    compat-libgcc-296              \
    compat-libstdc++-296           \
    compat-libstdc++-33            \
    gcc                            \
    gcc-c++                        \
    gnome-libs                     \
    gnome-libs-devel               \
    libaio-devel                   \
    libaio                         \
    make                           \
    openmotif21                    \
    xorg-x11-deprecated-libs-devel \
    xorg-x11-deprecated-libs
```

Many of these packages depend on other packages. For example, **`compat-gcc-32`** requires **`binutils`**, **`gcc`** and several other packages. You may have to install the following RPMs to satisfy Oracle's dependencies:

```
rpm -Uvh compat-db-4.1.25-9.i386.rpm     \
    compat-gcc-32-3.2.3-47.3.i386.rpm             \
    glibc-devel-2.3.4-2.i386.rpm                  \
    glibc-headers-2.3.4-2.i386.rpm                \
    glibc-kernheaders-2.4-9.1.87.i386.rpm         \
    cpp-3.4.3-9.EL4.i386.rpm                      \
    compat-gcc-32-c++-3.2.3-47.3.i386.rpm         \
    compat-libstdc++-33-3.2.3-47.3.i386.rpm       \
    gcc-3.4.3-9.EL4.i386.rpm                      \
    cpp-3.4.3-9.EL4.i386.rpm                      \
    gcc-c++-3.4.3-9.EL4.i386.rpm                  \
    libstdc++-devel-3.4.3-9.EL4.i386.rpm          \
    openmotif21-2.1.30-11.RHEL4.2.i386.rpm        \
    xorg-x11-deprecated-libs-6.8.1-23.EL.i386.rpm \
    compat-libgcc-296-2.96-132.7.2.i386.rpm       \
    compat-libstdc++-296-2.96-132.7.2.i386.rpm    \
    libaio-0.3.102-1.i386.rpm                     \
    libaio-devel-0.3.102-1.i386.rpm
```

The X.org development libraries, **`xorg-x11-deprecated-libs-devel`** and **`xorg-x11-devel`**, are required for the Oracle patch 4198954, the following RPMs may need to be installed:

```
rpm -Uvh xorg-x11-deprecated-libs-devel-6.8.1-23.EL.i386.rpm    \
    xorg-x11-devel-6.8.1-23.EL.i386.rpm                   \
    fontconfig-devel-2.2.3-7.i386.rpm                     \
    pkgconfig-0.15.0-3.i386.rpm                           \
    freetype-devel-2.1.9-1.i386.rpm                       \
    zlib-devel-1.2.1.2-1.i386.rpm
```

The Gnome libraries, **`gnome-libs`** and **`gnome-libs-devel`**, may require following RPMs to be installed:

```
rpm -Uvh gnome-libs-1.4.1.2.90-44.1.i386.rpm    \
```

```
    gnome-libs-devel-1.4.1.2.90-44.1.i386.rpm  \
    ORBit-0.5.17-14.i386.rpm                   \
    ORBit-devel-0.5.17-14.i386.rpm             \
    alsa-lib-1.0.6-4.i386.rpm                  \
    audiofile-0.2.6-1.i386.rpm                 \
    esound-0.2.35-2.i386.rpm                   \
    esound-devel-0.2.35-2.i386.rpm              \
    gtk+-1.2.10-33.i386.rpm                    \
    gtk+-devel-1.2.10-33.i386.rpm              \
    imlib-1.9.13-23.i386.rpm                   \
    imlib-devel-1.9.13-23.i386.rpm             \
    libpng10-1.0.16-1.i386.rpm                 \
    alsa-lib-devel-1.0.6-4.i386.rpm            \
    audiofile-devel-0.2.6-1.i386.rpm           \
    gdk-pixbuf-0.22.0-15.1.i386.rpm            \
    glib-devel-1.2.10-15.i386.rpm              \
    indent-2.2.9-6.i386.rpm                    \
    libjpeg-devel-6b-33.i386.rpm               \
    libtiff-devel-3.6.1-7.i386.rpm             \
    libungif-4.1.3-1.i386.rpm                  \
    libungif-devel-4.1.3-1.i386.rpm
```

Note

If you are using the Red Hat Network(RHN), you can run:

`up2date gnome-libs gnome-libs-devel`

You can use the **`up2date`** command for any packages. It takes care of dependencies by installing all required packages automatically.

To install the **`compat-oracle-rhel4`** and **`compat-libcwait`** packages you have to download the patch 4198954 from *http://metalink.oracle.com*. Make sure to select the Linux x86 platform. To unzip the downloaded **`p4198954_21_LINUX.zip`** file, run:

```
$ unzip p4198954_21_LINUX.zip
Archive:  p4198954_21_LINUX.zip
creating: 4198954/
inflating: 4198954/compat-oracle-rhel4-1.0-5.i386.rpm
inflating: 4198954/compat-libcwait-2.0-2.i386.rpm
inflating: 4198954/README.txt
```

Note

The **`compat-oracle-rhel4`** and **`compat-libcwait`** packages require the **`xorg-x11-deprecated-libs`** and **`xorg-x11-deprecated-libs-devel`** packages, see above. To install the two RPMs from the 4198954 patch, run:

`# rpm -Uvh 4198954/compat-oracle-rhel4-1.0-5.i386.rpm 4198954/compat-libcwait-2.0-2.i386.rpm`

Setting Up a Working Environment for Oracle

This chapter covers the creation of user accounts, directories and Oracle environments for your Oracle Database before you install.

32.1. Creating Oracle User Accounts

This section covers the command necessary to activate the required accounts to install an Oracle Database.

Create the group of users to be granted with SYSDBA system privilege

```
su - root groupadd dba
```

Now create a group owner for Oracle files

```
groupadd oinstall
useradd -c "Oracle software owner" -g oinstall -G dba oracle
passwd oracle
```

For more information on the "**oinstall**" group account, see *When to use OINSTALL*[1] group during install of oracle.

32.2. Creating Oracle Directories

Make sure that the Oracle file systems, in this example **/u01**, is large enough, see *Section 30.5, "Sizing Oracle Disk Space"* for more information.

```
su - root
#mkdir -p /u01/app/oracle/product/9.2.0
#chown -R oracle.oinstall /u01

#mkdir /var/opt/oracle
#chown oracle.dba /var/opt/oracle
#chmod 755 /var/opt/oracle
```

32.3. Setting Oracle Environment Variables

Make sure to set the following Oracle environment variables before you execute runInstaller.

As the oracle user execute the following commands: # Make sure to set the LD_ASSUME_KERNEL environment variable for Red Hat Enterprise Linux 3 and 4 !! # Use the "Linuxthreads with floating stacks" implementation instead of NPTL:

```
export LD_ASSUME_KERNEL=2.4.1    # for Red Hat Enterprise Linux 3
export LD_ASSUME_KERNEL=2.4.19   # for Red Hat Enterprise Linux 4
```

For the Oracle Environment

[1] https://metalink.oracle.com/oracleinstall/oracle8i/genericunix.html#Uoui

```
export ORACLE_BASE=/u01/app/oracle
export ORACLE_HOME=$ORACLE_BASE/product/9.2.0
export ORACLE_SID=test
export ORACLE_TERM=xterm
```

You will need to export *TNS_ADMIN= Set* if **sqlnet.ora**, **tnsnames.ora** and other variables are not present in **$ORACLE_HOME/network/admin**

```
export NLS_LANG=AMERICAN;
export ORA_NLS33=$ORACLE_HOME/ocommon/nls/admin/data
LD_LIBRARY_PATH=$ORACLE_HOME/lib:/lib:/usr/lib
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
export LD_LIBRARY_PATH
```

Set the shell search paths

```
export PATH=$PATH:$ORACLE_HOME/bin
```

You can add these environment settings to the end of the **~oracle/.bash_profile** file if you use bash. This will ensure that the environment variables are set permanently when you log in as "**oracle**", or when you switch to the user "**oracle**" by executing "**su - oracle**".

Starting runInstaller

An important point to note...

If you use CDs to install the database, do not change directory (**cd**) to **/mnt/cdrom** to execute **./runInstaller**! If you do so, the installation will fail because you will not be able to change the CDs.

Before you continue, make sure you have set the Oracle environment variables in *Section 32.3, "Setting Oracle Environment Variables"*.

Oracle no longer supports a character mode installer. Therefore, in order to execute runInstaller directly from a console of a machine you are logged into you need to set the DISPLAY environment variable. In the example the name of the node where Oracle is running is called "**oracleserver**". First, make sure that you also allow **runInstaller** on "**oracleserver**" to display X information on your Linux desktop machine. In the example, the computer name where you are running an X Windows desktop like KDE or GNOME is called "**yourdesktop**". Programs running on remote machines cannot display information to your screen unless you give them the authority to do so.

Before you execute **runInstaller**, execute **xterm** (or another terminal client of your choice) to see if your X setup is really working. If you are about to install Oracle on your desktop PC and not on a remote node, then you can skip step 1 and 3.

Step 1: Allow "**oracleserver**" to display X information to your desktop PC "**yourdesktop**":

```
yourdesktop:user$ xhost +oracleserver
```

Step 2: Open a new window and log in to the Oracle server "**oracleserver**" as the **root** user. This window will be used to **mount** and unmounting the Oracle CDs.

```
oracleserver:$ su - root
oracleserver:root# mount /mnt/cdrom
```

Step 3: From the console of your Oracle server "**oracleserver**" where you will run **runInstaller**, execute the following commands:

```
oracleserver:$ su - oracle
oracleserver:oracle$ export DISPLAY=yourdesktop:0.0
```

Step 4: Now you can execute **runInstaller** as "**oracle**" as shown in the next chapters. It is important that you do not change directory (**cd**) to **/mnt/cdrom**.

```
oracleserver:oracle$ /mnt/cdrom/runInstaller
```

Installing Oracle9i R2 (9.2.0.1.0) on Red Hat Advanced Server 2.1

You may get one or more errors during the Oracle installation. If you encounter a problem, read *Chapter 38, Oracle Installation Errors*. for more information and solutions. The errors can be addressed very easily using the prescribed solutions.

Installing Oracle9i R2 (9.2.0.4.0) on Red Hat Enterprise Linux 3

In order to install an Oracle9i R2 database on Red Hat Enterprise Linux 3, the "Oracle9iR2 Patch Set 3 9.2.0.4.0" patch set and a few other patches must be applied after the installation of Oracle9i Release 2 (9.2.0.1.0). Please note, there exists errors that can only be fixed by applying the 9.2.0.4 patch set.

35.1. Installing Oracle9i R2 (9.2.0.1.0) on Red Hat Enterprise Linux 3

Note

Throughout this chapter, the symbol # represents a terminal owned by the root user. You can use the root account by using the commands **su - root** or **sudo** if you have permissions.

Install the following required RPMs (read Oracle Note:252217.1 for more information) for compatibility:

```
# rpm -ivh      \
    compat-db-4.0.14-5.i386.rpm    \
    compat-gcc-7.3-2.96.122.i386.rpm   \
    compat-gcc-c++-7.3-2.96.122.i386.rpm   \
    compat-libstdc++-7.3-2.96.122.i386.rpm   \
    compat-libstdc++-devel-7.3-2.96.122.i386.rpm  \
    openmotif21-2.1.30-8.i386.rpm    \
    setarch-1.3-1.i386.rpm     \
    tcl-8.3.5-92.i386.rpm
```

Relink **gcc** so that the older **gcc** will be used during the Oracle installation (see Oracle Note:252217.1 for more information):

```
su - root
# mv /usr/bin/gcc /usr/bin/gcc323
# ln -s /usr/bin/gcc296 /usr/bin/gcc
# mv /usr/bin/g++ /usr/bin/g++323
# ln -s /usr/bin/g++296 /usr/bin/g++
```

Please Note

If you received an error stating g++ does not exist after executing the above commands, then gcc-c++ has not been installed.

When you execute **runInstaller** from the Oracle9i R2 (9.2.0) CD, you will get the following error message:

```
Error occurred during initialization of VM
Unable to load native library: /tmp/OraInstall2003-10-25_03-14-57PM/jre/lib/i386/libjava.so:
symbol __libc_wait, version GLIBC_2.0 not defined in file libc.so.6 with link time reference
```

To resolve the **__libc_wait** symbol issue, download the p3006854_9204 patch **p3006854_9204_LINUX.zip** from *http://metalink.oracle.com*. See bug 3006854 for more information. To apply the patch, run

```
# unzip p3006854_9204_LINUX.zip
Archive:  p3006854_9204_LINUX.zip
creating: 3006854/
inflating: 3006854/rhel3_pre_install.sh
inflating: 3006854/README.txt

# cd 3006854
# sh rhel3_pre_install.sh
Applying patch...
Patch successfully applied
#
```

If everything goes wrong...

If you get the following error when you run **rhel3_pre_install.sh**:

rhel3_pre_install.sh: line 36: gcc: command not found

The error means you must install or link **gcc** correctly. This will mean you cannot start any binaries any more, for example:

 # ls

ls: error while loading shared libraries: /etc/libcwait.so: cannot open shared object

file: No such file or directory

 # rm /etc/ld.so.preload

rm: error while loading shared libraries: /etc/libcwait.so: cannot open shared object

file: No such file or directory

#

To fix that, run the **echo**. echo is a built-in shell command and not a binary so it will still work.

 # echo "" > /etc/ld.so.preload rm /etc/ld.so.preload

Now you can start the process over from the beginning of the chapter.

Now **runInstaller** can be started from the CD:

```
su - oracle
$ echo $LD_ASSUME_KERNEL    # it is important that this variable is set!
2.4.1
$ /mnt/cdrom/runInstaller
```

- Welcome Screen: Click **Next**

- Inventory Location: Click **Next**

- Unix Group Name: Use "oinstall"

Click Next When asked to run **`/tmp/orainstRoot.sh`**, run it before you click **Continue**

- File Locations: Use the default values

- Available Products: Select "Oracle9i Database 9.2.0.1.0"

- Installation Types: Select **`Custom`** since we only want to install the software for now

- Available Products: Click **Next** or add some more components.

- Components Locations: Accept default values and click **Next**

- Privileged Operating System Groups: You can use the default values: OSDBA Group = dba, OSOPER Group = dba

- Oracle Managent Server Repository: You can use the default choice

- Create database: Select **`No`** since we first have to patch Oracle before a database can be created.

- Summary: Start the Install

- Configuration tools: The tools won't come up. Simply ignore it.

- At the end of the installation, exit **`runInstaller`**.

You may get the following errors while using the installer.

"Error in invoking target install of **`makefile /u01/app/oracle/product/9.2.0/network/lib/ins_oemagent.mk`**."

The **`/u01/app/oracle/product/9.2.0/install/make.log`** file reads:

```
/u01/app/oracle/product/9.2.0/network/lib/libnmi.a(snmitcln.o)(.text+0xa4e): In function
 `Nls_FormatCmd':
: undefined reference to `__ctype_b'
/u01/app/oracle/product/9.2.0/network/lib/libnmi.a(snmitcln.o)(.text+0x159d): In function
 `Nls_ScanCmd':
: undefined reference to `__ctype_b'
/u01/app/oracle/product/9.2.0/network/lib/libnmi.a(snmitcln.o)(.text+0x1603): more undefined
 references to `__ctype_b' follow
collect2: ld returned 1 exit status
make: *** [dbsnmp] Error 1
```

Click **ignore**.

This will be fixed after you apply the patch 3119415 after the 9.2.0.4 patchset has been applied. You can not apply the patch 3119415 at this time since the file **`/u01/app/oracle/oraInventory/ContentsXML/comps.xml`** does not exist yet. We will show you how to apply patch 3119415 later.

"Error in invoking target install of **`makefile /u01/app/oracle/product/9.2.0/ctx/lib/ins_ctx.mk`**. "

The /u01/app/oracle/product/9.2.0/install/make.log file reads:

```
/usr/bin/ld: ctxhx: hidden symbol `stat' in /usr/lib/libc_nonshared.a(stat.oS) is referenced
 by DSO
collect2: ld returned 1 exit status
make: *** [ctxhx] Error 1
```

Click **ignore**. This will be fixed when you apply the 9.2.0.4 patch set.

35.2. Patching Oracle9i to 9.2.0.4.0 on Red Hat Enterprise Linux 3

To patch Oracle9i R2, download the Oracle9i Release 2 Patch Set 3 Version 9.2.0.4.0 for Linux x86 from *http://metalink.oracle.com*. Download the **p3095277_9204_LINUX.zip** file to **/tmp** and run the following command:

```
su - oracle
$ cp p3095277_9204_LINUX.zip /tmp
$ cd /tmp
$ unzip p3095277_9204_LINUX.zip
Archive:  p3095277_9204_LINUX.zip
inflating: 9204_lnx32_release.cpio
inflating: README.html
inflating: patchnote.css
$
$ cpio -idmv < 9204_lnx32_release.cpio
Disk1/stage/locks
Disk1/stage/Patches/oracle.apache.isqlplus/9.2.0.4.0/1/DataFiles/bin.1.1.jar
Disk1/stage/Patches/oracle.apache.isqlplus/9.2.0.4.0/1/DataFiles/lib.1.1.jar
...
```

To patch the **runInstaller**, execute:

```
su - oracle
$ echo $LD_ASSUME_KERNEL    # it is important that this variable is set!
2.4.1
$ cd /tmp/Disk1/
$ ./runInstaller
```

- Welcome Screen: Click **Next**

- File Locations: Use default values

- Available Products: Select "Oracle Universal Installer 2.2.0.18.0"

- Components Locations: Accept default values and click **Next**

- Summary: Start the Install

- At the end of the installation, you must exit **runInstaller**

To patch Oracle9i R2, execute:

```
su - oracle
$ echo $LD_ASSUME_KERNEL    # it is important that this variable is set!
2.4.1
$ cd $ORACLE_HOME/bin
$ ./runInstaller
```

- Welcome Screen: Click **Next**

- File Locations: Use default values

- Available Products: Select "Oracle9iR2 Patch Set 3 9.2.0.4.0 !"

- Summary: Start the Install

- At the end of the installation, exit **runInstaller**

You may get the error: "Error in invoking target install of **makefile /u01/app/oracle/product/9.2.0/network/lib/ins_oemagent.mk**".

The **/u01/app/oracle/product/9.2.0/install/make.log** file reads:

```
/u01/app/oracle/product/9.2.0/network/lib/libnmi.a(snmitcl.o)(.text+0x1cc): In function
 `get_ora_stmt_handle':
: undefined reference to `__ctype_b'
/u01/app/oracle/product/9.2.0/network/lib/libnmi.a(snmitcl.o)(.text+0x124e): In function
 `OraProcess_Oid':
: undefined reference to `__ctype_b'
/u01/app/oracle/product/9.2.0/network/lib/libnmi.a(snmitcl.o)(.text+0x176c): more undefined
 references to `__ctype_b' follow
collect2: ld returned 1 exit status
make: *** [dbsnmp] Error 1
```

Click ignore. This will be fixed by applying the patch 3119415 after the 9.2.0.4 patchset has been applied. The patch 3119415 cannot be applied while the patch process for the 9.2.0.4 patchset is running.

After the 9.2.0.4 patchset has been applied, download the patch **p3119415_9204_LINUX.zip** from *http://metalink.oracle.com*. See bug 3119415 for more information. Also, download the **opatch Release 2.2.0** utility from *http://metalink.oracle.com*. See bug 2617419 at *http://metalink.oracle.com* for more information.

To install **opatch**, run:

```
su - oracle
$ cp p2617419_210_GENERIC.zip /tmp
$ cd /tmp
$ unzip p2617419_210_GENERIC.zip
```

Before you apply the 3119415 patch, you need to make sure the fuser binary can be found by the oracle user, see the **PATH** environment variable below. Otherwise the patch can not be applied because the **fuser** binary is used by **opatch**. To apply the 3119415 patch, run

```
su - oracle
$ unzip p3119415_9204_LINUX.zip
$ cd 3119415
$ export PATH=$PATH:/tmp/OPatch
$ export PATH=$PATH:/sbin        # the patch needs "fuser" which is located in /sbin
$ which opatch
/tmp/OPatch/opatch
$ opatch apply
```

Now you should be able to create a database with dbca:

```
$ su - oracle
```

```
$ dbca
```

35.3. Patching Oracle Intelligent Agent on Red Hat Enterprise Linux 3

When you run "**agentctl start**" (Oracle 9.2.0.4), **dbsnmp** will crash:

```
$ su - oracle
$ agentctl start

DBSNMP for Linux: Version 9.2.0.4.0 - Production on 07-JAN-2004 19:11:14

Copyright (c) 2003 Oracle Corporation.  All rights reserved.

Starting Oracle Intelligent Agent.../u01/app/oracle/product/9.2.0/bin/dbsnmpwd: line 156:
 1855 Segmentation fault      nohup $ORACLE_HOME/bin/dbsnmp $*
>>$DBSNMP_WDLOGFILE 2>&1
/u01/app/oracle/product/9.2.0/bin/dbsnmpwd: line 156:  1868 Segmentation fault      nohup
 $ORACLE_HOME/bin/dbsnmp $* >>$DBSNMP_WDLOGFILE 2>&1
/u01/app/oracle/product/9.2.0/bin/dbsnmpwd: line 156:  1880 Segmentation fault      nohup
 $ORACLE_HOME/bin/dbsnmp $* >>$DBSNMP_WDLOGFILE 2>&1
/u01/app/oracle/product/9.2.0/bin/dbsnmpwd: line 156:  1892 Segmentation fault      nohup
 $ORACLE_HOME/bin/dbsnmp $* >>$DBSNMP_WDLOGFILE 2>&1
```

To resolve this problem, apply the patch **p3238244_9204_LINUX.zip** from *http://metalink.oracle.com*, search for 3238244 to see the bug and patch information.

Before you apply the patch, make sure the instance is down!

Make sure the **opatch** script appears in your **$PATH**. To verify if **opatch** is in your **$PATH**, run the following commands:

```
$ su - oracle
$ which opatch
/tmp/OPatch/opatch
$
```

Please note, the patch needs "**fuser**" which should be located in **/sbin**, if it is not you will need to install it. To apply now the patch, run:

```
$ su - oracle
$ unzip p3238244_9204_LINUX.zip
$ cd 3238244
$ export PATH=$PATH:/sbin
$ opatch apply
```

Now you need to relink **dbsnmp**. This is the binary that crashed when running **agentctl start**. To find which **makefile** handles the linking of **dbsnmp**, you can run:

```
$ su - oracle
$ find $ORACLE_HOME -name "*.mk" | xargs grep -l dbsnmp
/u01/app/oracle/product/9.2.0/network/lib/ins_oemagent.mk
/u01/app/oracle/product/9.2.0/network/lib/env_oemagent.mk
$
```

To relink **dbsnmp** and all associated executable files which are maintained by the **ins_oemagent.mk makefile**, run:

```
$ su - oracle
$ cd $ORACLE_HOME/network/lib
$ make -f ins_oemagent.mk install
```

Now you should be able to start the agent:

```
$ su - oracle
$ agentctl start
```

Note

Do not forget to undo the links to **`/usr/bin/gcc and /usr/bin/g++`** if you do not need them any more. You may also want to undo the changes in **`/etc/ld.so.preload`** file.

Installing Oracle9i R2 (9.2.0.6.0) on Red Hat Enterprise Linux 4

In order to install Oracle9i Release 2 (9.2.0.6), the 9.2.0.6 patch set must be applied for the Oracle database server (patch number 3948480) after the Oracle9i Release 2 (9.2.0.4) installation. For more information, see Oracle9i Release Notes Release 2 (9.2.0.4.0) for Linux x86 - Red Hat Enterprise Linux 4 Certification Update.

36.1. Installing Oracle9i R2 (9.2.0.4.0) on Red Hat Enterprise Linux 4

Before you continue, ensure all the required RPMs are installed, see Checking Packages (RPMs). Also ensure LD_ASSUME_KERNEL is set to 2.4.19 (see Setting Oracle Environments):

```
$ su - oracle
$ echo $LD_ASSUME_KERNEL
2.4.19
$
```

Now execute **`runInstaller`**:

```
$ su - oracle
$ echo $LD_ASSUME_KERNEL
2.4.19
$ /media/cdrom/runInstaller
```

- Welcome Screen: Click **Next**

- Inventory Location: Click OK

- Unix Group Name: Use "oinstall" and click **Next**. When asked to run **`/tmp/orainstRoot.sh`** enter the following into the command line

```
/tmp/orainstRoot.sh
```

Then click **Continue**

- File Locations: Use default values

- Available Products: Select "**Oracle9i Database 9.2.0.4.0**"

- Installation Types: Select **Custom** since we only want to install the software for now

- Available Products: Click **Next** or add some more components and then click **Next**. - Components Locations: Accept the default values and click **Next**

- Privileged Operating System Groups: Use the default values: **`OSDBA Group = dba, OSOPER Group = dba`**

- Oracle Managent Server Repository: Use the default choice

- Create database: Select **NO** since we first need to patch the Oracle database software

- Summary: Start the Install

36.2. Patching Oracle9i R2 to 9.2.0.6.0 on Red Hat Enterprise Linux 4

Download the patch 3948480, Oracle9i Patch Set Release 2 (9.2.0.6) Patch Set 5, from *http://metalink.oracle.com* and execute the following commands to extract it:

```
su - oracle
$ cp p3948480_9206_LINUX.zip /tmp
$ cd /tmp
$ unzip p3948480_9206_LINUX.zip
Archive:  p3948480_9206_LINUX.zip
creating: Disk1/
creating: Disk1/stage/
creating: Disk1/stage/Patches/
...
```

Now download the patch 4188455 from *http://metalink.oracle.com*. This patch is needed for launching the **`runInstaller`** that came with the patch 3948480 we just downloaded above.

```
su - oracle
$ cp p4188455_10103_LINUX.zip /tmp
$ cd /tmp
$ unzip p4188455_10103_LINUX.zip
Archive:  p4188455_10103_LINUX.zip
inflating: oraparam.ini
inflating: README.txt
$
```

The **`/tmp/oraparam.ini`** file will now be used for launching the **`runInstaller`** that came with the patch 3948480.

To patch the **`runInstaller`** application, run:

```
su - oracle
$ echo $LD_ASSUME_KERNEL
2.4.19
$ /tmp/Disk1/install/runInstaller   - paramFile  /tmp/oraparam.ini
```

- Welcome Screen: Click **Next**

- File Locations: Use default values (in this example: **`/tmp/Disk1/stage/products.xml`**)

- Available Products: Select "**Oracle Universal Installer 10.1.0.3.0**"

- Summary: Click **Install**

- At the end of the installation, you must exit runInstaller

Ensure that no Oracle processes are running:

```
ps -ef | grep ora
```

Now patch Oracle9i R2:

```
su - oracle
$ echo $LD_ASSUME_KERNEL    # it is important that this variable is set!
2.4.19
$ /tmp/Disk1/install/runInstaller -paramFile /tmp/oraparam.ini
```

- Welcome Screen: Click **Next**

- File Locations: Use default values (in this example: **`/tmp/Disk1/stage/products.xml`**)

- Available Products: Select "**Oracle 9iR2 Patchset 9.2.0.6.0**"

- Summary: Click **Install**

When you are asked to run **`root.sh`**, run it before you click **Continue**

- At the end of the installation, exit **runInstaller**.

After the 9.2.0.6 patchset has been applied, download the patch 4190568 from *http://metalink.oracle.com*. Download the **`opatch`** utility for release 10.1.0.2 (patch 2617419) from *http://metalink.oracle.com*.

To install **`opatch`**, run:

```
su - oracle
$ cp p2617419_10102_GENERIC.zip /tmp
$ cd /tmp
$ unzip p2617419_10102_GENERIC.zip
$ cp -a /tmp/OPatch/ $ORACLE_HOME
```

To apply the 4190568 patch, run

```
su - oracle
$ unzip p4190568_9206_LINUX.zip
$ cd 4193454
$ export PATH=$PATH:$ORACLE_HOME/OPatch
$ opatch apply
```

If you intend to use Direct I/O Support, you must also download and apply patch 2448994.

Now you should be able to create a database with dbca:

```
$ su - oracle
$ dbca
```

If **`dbca`** dies on the system with the following error:

```
/u01/app/oracle/product/9.2.0/bin/dbca: line 124: 26649 Segmentation fault
$JRE_DIR/bin/jre -DORACLE_HOME=$OH -DJDBC_PROTOCOL=thin -mx64m -classpath $CLASSPATH
 oracle.sysman.assistants.dbca.Dbca $ARGUMENTS
```

You can execute the following command:

```
su - root
touch /etc/rac_on
```

Now you can restart the process to get **`dbca`** to work.

Starting and Shutting down the Oracle9i Database

This chapter shows two separate ways to start and shut down an Oracle9i Database; Using **SQLplus** and using the **ORACLE_HOME** binaries.

sqlplus

Note

In Oracle9i **svrmgrl** is no longer supported, however, you can now do everything with sqlplus so nothing is lost.

To start up the database, run the following commands:

```
oracle$ sqlplus /nolog
SQL> connect / as sysdba
SQL> startup
```

The slash connects you to the schema owned by SYS. So in this example you will be connected to the schema owned by SYS with the privilege **SYSDBA**. SYSDBA gives you the following privileges:

- **sysoper** privileges WITH ADMIN OPTION
- create databases
- recover one or more databases to a point

Using the Oracle binaries dbstart and dbstop

Tip

The **dbstart** and **dbstop** commands are in the **$ORACLE_HOME/bin/dbstart** and **$ORACLE_HOME/bin/dbshut** directories respectively.

You can use **$ORACLE_HOME/bin/dbstart** to start up the database, and **$ORACLE_HOME/bin/dbshut** to shut down the database. To get **$ORACLE_HOME/bin/dbstart** and **$ORACLE_HOME/bin/dbshut** working, you need to change the third field for your Oracle **SID** in **/etc/oratab** from "**N**" to "**Y**".

No to read:

```
test:/u01/app/oracle/product/9.2.0:N
```

Yes to read:

```
test:/u01/app/oracle/product/9.2.0:Y
```

In some cases you may have to copy the init file for your SID (in this example "**test**") from **/u01/app/oracle/admin/test/pfile** to **$ORACLE_HOME/dbs** to get **dbstart** and **dbshut** working. In the Below command remember to change **test** to something appropriate.

```
cp /u01/app/oracle/admin/test/pfile/inittest.ora.642002224936 \
$ORACLE_HOME/dbs/inittest.ora
```

Then make sure that your init file already exists in **$ORACLE_HOME/dbs**.

Oracle Installation Errors

This chapter covers many common errors and pitfalls associated with installing an Oracle9i Database.

Log Files

Always check first the error logs for 9.2.0 in **`/tmp/OraInstall*`**, where "*" will be the date of the install, for example **`/tmp/OraInstall2002-07-04_09-50-19PM`**. When you problems with **`make`**, see also the **`$ORACLE_HOME/install/make.log`** file.

`make` Problems

First ensure that gcc is installed on your system by executing:

```
$ which gcc
/usr/bin/gcc
```

Here is the command to find the RPM package name for **`/usr/bin/gcc`**:

```
$ rpm -qf /usr/bin/gcc
gcc-2.96-98
```

Verify that your error is not one of the other error messages below. See *Chapter 31, Verifying Required Packages(RPMs)* for more information on whether you have the correct packages.

Error in invoking target install of makefile/u01/app/oracle/product/9.2.0/ctx/lib/ins_ctx.mk

You may see the following errors in **`$ORACLE_HOME/install/make.log`**:

```
/lib/libdl.so.2: undefined reference to `_dl_addr@GLIBC_PRIVATE'
/lib/libdl.so.2: undefined reference to `_dl_open@GLIBC_PRIVATE'
/lib/libdl.so.2: undefined reference to `_dl_close@GLIBC_PRIVATE'
/lib/libdl.so.2: undefined reference to `_dl_sym@GLIBC_PRIVATE'
/lib/libdl.so.2: undefined reference to `_dl_vsym@GLIBC_PRIVATE'
```

This error comes up when the following step is executed:

```
/usr/bin/make -f ins_ctx.mk install ORACLE_HOME=/u01/app/oracle/product/9.2.0
```

Edit the file **`$ORACLE_HOME/ctx/lib/env_ctx.mk`**, add "**`$(LDLIBFLAG)dl`**" to the "**`INSO_LINK =`**" line. The updated line should with the added "**`$(LDLIBFLAG)dl`**" flag, should look like this:

```
INSO_LINK = -L$(CTXLIB) $(LDLIBFLAG)m $(LDLIBFLAG)dl $(LDLIBFLAG)sc_ca $(LDLIBFLAG)sc_fa
 $(LDLIBFLAG)sc_ex $(LDLIBFLAG)sc_da $(LDLIBFLAG)sc_ut $(LDLIBFLAG)sc_ch $(LDLIBFLAG)sc_fi
 $(LLIBCTXHX) $(LDLIBFLAG)c -Wl,-rpath,$(CTXHOME)lib $(CORELIBS) $(COMPEOBJS)
```

After that hit **Retry** in the error pop-up.

If this did not fix the problem, try the following solution:

Edit the file **`$ORACLE_HOME/ctx/lib/env_ctx.mk`** again, go to "**`INSO_LINK =`**", remove the above entry you made and add "**`` `cat $(LIBHOME)/sysliblist` ``**" to the line and save it. This is what the updated line, with the added "**`` `cat $(LIBHOME)/sysliblist` ``**" string, look like:

```
INSO_LINK = -L$(CTXLIB) $(LDLIBFLAG)m `cat $(LIBHOME)/sysliblist` $(LDLIBFLAG)sc_ca
 $(LDLIBFLAG)sc_fa $(LDLIBFLAG)sc_ex $(LDLIBFLAG)sc_da $(LDLIBFLAG)sc_ut $(LDLIBFLAG)sc_ch
 $(LDLIBFLAG)sc_fi $(LLIBCTXHX) $(LDLIBFLAG)c -Wl,-rpath,$(CTXHOME)lib $(CORELIBS)
 $(COMPEOBJS)
```

After that hit **Retry** in the error pop-up.

ORA-27123: unable to attach to shared memory segment

This error, "ORA-27123: unable to attach to shared memory segment", message may came up when the **Oracle Database Configuration Assistant** was running. Execute the following command to temporarily increase the maximum shared memory size:

```
su - root
# cat /proc/sys/kernel/shmmax
33554432
# echo `expr 1024 \* 1024 \* 1024` > /proc/sys/kernel/shmmax
# cat /proc/sys/kernel/shmmax
1073741824
#
```

Then click **Retry** for the **Oracle Database Configuration Assistant**.

It is recommended to increase the shmmax setting permanently for Oracle9i. So if you want to increase the maximum shared memory size permanently, add the following line to the **`/etc/sysctl.conf`** file:

```
kernel.shmmax=1073741824
```

For more information on setting shared memory parameters for Oracle, see *Chapter 7, Setting Shared Memory*.

ORA-03113: end-of-file on communication channel

You may see this error, ORA-03113: end-of-file on communication channel, when you run the "**Database Configuration Assistant**" and "**`sqlplus`**". It can be caused by **`shmmax`** parameter being too small. Make sure to increase **`shmmax`** permanently. Read *Chapter 7, Setting Shared Memory* for information on how to increase the **`shmmax`** parameter.

Error in invoking target install of make file /u01/app/oracle/product/9.2.0/network/lib/ins_oemagent.mk

If you see the error, Error in invoking target install of make file /u01/app/oracle/product/9.2.0/network/lib/ins_oemagent.mk, on Red Hat Enterprise Linux 3, follow the guide lines at *Chapter 35, Installing Oracle9i R2 (9.2.0.4.0) on Red Hat Enterprise Linux 3*.

An error when executing "`agentctl start`"

The error below may have occurred when "**`agentctl start`**" is executed.

```
$ agentctl start

DBSNMP for Linux: Version 9.2.0.4.0 - Production on 07-JAN-2004 19:11:14
```

```
Copyright (c) 2003 Oracle Corporation. All rights reserved.

Starting Oracle Intelligent Agent.../u01/app/oracle/product/9.2.0/bin/dbsnmpwd:
line 156: 1855 Segmentation fault nohup $ORACLE_HOME/bin/dbsnmp $*
>>$DBSNMP_WDLOGFILE 2>&1
/u01/app/oracle/product/9.2.0/bin/dbsnmpwd: line 156: 1868 Segmentation fault
nohup $ORACLE_HOME/bin/dbsnmp $* >>$DBSNMP_WDLOGFILE 2>&1
/u01/app/oracle/product/9.2.0/bin/dbsnmpwd: line 156: 1880 Segmentation fault
nohup $ORACLE_HOME/bin/dbsnmp $* >>$DBSNMP_WDLOGFILE 2>&1
/u01/app/oracle/product/9.2.0/bin/dbsnmpwd: line 156: 1892
```

You are probably trying to start the agent on Red Hat Enterprise Linux 3. See *Section 35.3, "Patching Oracle Intelligent Agent on Red Hat Enterprise Linux 3"*for how to resolve it.

Errors in dbca

If you receive the error:

```
$ dbca
SIGSEGV   11*  segmentation violation
        stackbase=0x453da000, stackpointer=0x453d9d5c
Full thread dump:
    "AWT-EventQueue-0" (TID:0x411d1e20, sys_thread_t:0x453d9e0c,
state:R) prio=5 *current thread*
        java.lang.Object.wait(Object.java)
        java.awt.EventQueue.getNextEvent(EventQueue.java:126)
...
```

or

```
/u01/app/oracle/product/9.2.0/bin/dbca: line 124: 26649 Segmentation fault
$JRE_DIR/bin/jre -DORACLE_HOME=$OH -DJDBC_PROTOCOL=thin -mx64m -classpath
$CLASSPATH oracle.sysman.assistants.dbca.Dbca $ARGUMENTS
```

If this happens, try the following:

```
$ su - root
touch /etc/rac_on
```

Now try to restart **dbca**.

Another option is to edit **$ORACLE_HOME/bin/dbca** and to put the following lines under comment except the line not preceded by a hash:

```
# if [ -f /etc/rac_on ]; then
# Run DBCA
$JRE_DIR/bin/jre -native -DORACLE_HOME=$OH ...
# else
# Run DBCA
# $JRE_DIR/bin/jre -DORACLE_HOME=$OH ...
# fi
```

Now try to restart dbca.

gcc errors in Red Hat Enterprise Linux 3

When installing or running Oracle on Red Hat Enterprise Linux 3 you may encounter this gcc error.

```
gcc -o /u01/app/oracle/product/9.2.0/rdbms/lib/oracle \
 -L/u01/app/oracle/product/9.2.0/rdbms/lib/ ...
...
/usr/bin/ld: /u01/app/oracle/product/9.2.0/rdbms/lib/oracle: hidden symbol `__fixunssfdi' in /
usr/lib/gcc-lib/i386-redhat-linux/3.2.3/libgcc.a(_fixunssfdi.oS) is referenced by DSO

collect2: ld returned 1 exit status
make: *** [/u01/app/oracle/product/9.2.0/rdbms/lib/oracle] Error 1
/usr/bin/make -f ins_rdbms.mk ioracle ORACLE_HOME=/u01/app/oracle/product/9.2.0
```

To fix the linking problem, execute the following commands:

```
# mv /usr/bin/gcc /usr/bin/gcc323
# mv /usr/bin/g++ /usr/bin/g++323
# ln -s /usr/bin/gcc296 /usr/bin/gcc
# ln -s /usr/bin/g++296 /usr/bin/g++
```

Now you should be able to relink the oracle binary again.

Once you are done, you may want to undo the changes you have performed above:

```
# mv /usr/bin/gcc323 /usr/bin/gcc
# mv /usr/bin/g++323 /usr/bin/g++
```

./runInstaller: line 58: ./runInstaller: cannot execute binary file

You are probably trying to run a 64 bit Oracle version on a 32 bit Linux system. Make sure you downloaded the right Oracle version for your Linux system.

To check if runInstaller is a 32 bit binary or a 64 bit binary, run the following command:

```
$ cd /mnt/cdrom
$ file install/linux/runInstaller
install/linux/runInstaller: ELF 32 bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/
Linux 2.0.0, dynamically linked (uses shared libs), not stripped
```

To check if your Linux system is 32 bit system or a 64 bit system, run e.g. the following command:

```
$ file /sbin/init
/sbin/init: ELF 32 bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.2.5,
 dynamically linked (uses shared libs), not stripped
```

The Oracle installer runInstaller Hangs

The Oracle installer runInstaller hangs at:

```
Installing Java Runtime Environment... Link pending... Copying README...
```

You may encounter this problem on Red Hat Enterprise Linux 3. You probably forgot to set the environment variable **`LD_ASSUME_KERNEL`** to **`2.4.1`**.

To rectify this problem, run the following command and then restart **`runInstaller`**:

```
oracle$ export LD_ASSUME_KERNEL=2.4.1
```

Recovery Manager(rman) Hangs

You are probably running the wrong **rman** binary from the **XFree86-devel** RPM:

```
$ which rman
/usr/X11R6/bin/rman
```

Can't find init file for Database "SID"

You may see this error when you start the database with dbstart.

Copy the init file for your SID (in this example “test”) from from **/u01/app/oracle/admin/test/pfile** to **$ORACLE_HOME/dbs** to get **dbstart** and **dbshut** working:

```
cp /u01/app/oracle/admin/test/pfile/inittest.ora.642002224936 $ORACLE_HOME/dbs/inittest.ora
```

Error in setting permissions of file/directory /u01/app/oracle/jre/1.1.8/bin/i686/native_threads/.extract_args

This may happen if the CD was not burned correctly. The solution is to check the image checksums and then burn the cd again or download the image again.

Various Oracle SID Errors

If you get the error ORA-01034: ORACLE not available, ORA-27101: shared memory realm does not exist, Linux Error: 2: No such file or directory or ORA-01034: ORACLE not available then check if ORACLE_SID is set correctly.

If **ORACLE_SID** is set correctly, then you probably have a trailing slash "/" on the **ORACLE_HOME** environment variable. Remove it by changing **ORACLE_HOME=/u01/app/oracle/product/9.2.0/** to **ORACLE_HOME=/u01/app/oracle/product/9.2.0** and try again to connect to **sys**. Remember the **ORACLE_HOME** parameter may be slightly different on your machine.

jre was not found in /tmp/OraInstall/jre/bin/i586/green_threads/jre

You are probably running **runInstaller** on a 586 machine, or your AMD CPU is falsely recognized as 586 (the case for a AMD K6-III-400). You can check your machine (hardware) type by executing "uname -m". If you are not running on a 586 or on a AMD machine, try to link **jre** to java and see if this solves your problem.

To rectify the problem with the 586 machine or with the AMD CPU, create a link for lib and bin from i586 to i686 and make the i686 directories read only. For example:

```
ln -s /tmp/OraInstall/jre/bin/i686 /tmp/OraInstall/jre/bin/i586
ln -s /tmp/OraInstall/jre/lib/i686 /tmp/OraInstall/jre/lib/i586
chmod u-w /tmp/OraInstall/jre/bin/i686/tmp/OraInstall/jre/lib/i686
```

Now restart **runInstaller**.

./jre/bin/i386/native_threads/java: error while loading shared libraries: libstdc++-libc6.1-1.so.2: cannot open shared object file: No such file or directory

You probably forgot to install the **compat-libstdc++** RPM, the package for "Standard C++ libraries for Red Hat Linux 6.2 backwards compatibility". To rectify this problem, install the **compat-libstdc++** RPM. Read *Chapter 31, Verifying Required Packages(RPMs)* for more information.

/u01/app/oracle/jre/1.1.8/bin/../lib/i686/green_threads/libzip.so: symbol errno, version GLIBC_2.0 not defined in file libc.so.6 with link time reference (libzip.so) Unable to initialize threads: cannot find class java/lang/Thread Could not create Java VM

You may experience this problem when running the Database Configuration Assistant dbca on Red Hat Enterprise Linux 3 but forgot to set the LD_ASSUME_KERNEL environment variable.

To rectify this problem, run the following command as the **oracle** user on Red Hat Enterprise Linux 3 and restart **dbca**:

```
oracle$ export LD_ASSUME_KERNEL=2.4.1
```

lsnrctl status or lsnrctl start Errors

When **lsnrctl status** or **lsnrctl start** are executed you may get the following output:

```
$ lsnrctl status (or lsnrctl start)
LSNRCTL for Linux: Version 9.2.0.4.0 - Production on 14-OCT-2004 14:33:10
Copyright (c) 1991, 2002, Oracle Corporation. All rights reserved.
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC))) TNS-12541: TNS:no listener
TNS-12560: TNS:protocol adapter error
TNS-00511: No listener
Linux Error: 2: No such file or directory
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=xxxx)(PORT=1521)))
TNS-12541: TNS:no listener
TNS-12560: TNS:protocol adapter error
TNS-00511: No listener
Linux Error: 111: Connection refused
```

One of the possibilities are that the **/var/tmp/.oracle** directory does not exist. If that is the case, run the following commands:

```
su - root
# mkdir /var/tmp/.oracle
# chown oracle:dba /var/tmp/.oracle
```

Now try to run **lsnrctl start** as **oracle** again.

Java Cannot Connect to X11

First ensure you followed the instructions in *Chapter 33, Starting runInstaller* very closely.

If you get this error:

```
Exception in thread "main" java.lang.InternalError: Can't connect to X11 window server using
 'alpha:0.0' as the value of the DISPLAY variable.
        at sun.awt.X11GraphicsEnvironment.initDisplay(Native Method)
        at sun.awt.X11GraphicsEnvironment.(X11GraphicsEnvironment.java:59)
        at java.lang.Class.forName0(Native Method)
        at java.lang.Class.forName(Class.java:120)
        at
 java.awt.GraphicsEnvironment.getLocalGraphicsEnvironment(GraphicsEnvironment.java:58)
        at java.awt.Window.(Window.java:188)
        at java.awt.Frame.(Frame.java:315)
        at java.awt.Frame.(Frame.java:262)
        at oracle.sysman.oii.oiic.OiicInstaller.main(OiicInstaller.java:593)
```

Note

If you use newer Red Hat Enterprise Linux versions as your desktop and you want to install the database on another machine, then you need to set the **`DisallowTCP`** entry in **`/etc/X11/gdm/gdm.conf`** for the GNOME Display Manager to read:

```
DisallowTCP=false
```

After that you need to restart your X server. You can do this runnning the init command as root:

```
# init 3# init 5
```

Other Errors

For other errors, issues and problems your best bet is to search the *Oracle on Linux Discussion Forum*[1].

[1] http://forums.oracle.com/forums/forum.jspa?forumID=135

Reference List

puschitz.com[1]

Oracle Database 10g Release 1 (10.1) Documentation[2]

Oracle Database 10g Release 2 (10.2) Documentation [3]

Oracle Database 10g Linux Administration [4]

http://www.bookpool.com/sm/0072230533

GFS and Oracle 10gR2 RAC Installation[5]

GFS and Oracle 10gR2 RAC Installation (PDF)[6]

Red Hat Enterprise Linux AS 4 Release Notes [7]

Upgrading from Red Hat Enterprise Linux 2.1 AS To Red Hat Enterprise Linux 3 [8]

An Overview of Red Hat Advanced Server V2.1 Reliability, Availability, Scalability, and Manageability (RASM) Features [9]

Linux Virtual Memory in Red Hat Advanced Server 2.1 and Oracle's Memory Usage Characteristics [10]

Oracle9iR2 on Linux: Performance, Reliability and Manageability Enhancements on Red Hat Linux Advanced Server 2.1[11]

Understanding Virtual Memory. [12]

Understanding the Linux Kernel [13]

High Memory In The Linux Kernel [14]

Optimizing Linux I/O [15]

To read the Oracle MetaLink links refer to *http://metalink.oracle.com* and input the numbers. Note that registration is required.

Oracle MetaLink Note:200266.1

Oracle MetaLink Note:225751.1

[1] http://www.puschitz.com/
[2] http://www.oracle.com/technology/documentation/database10g.html
[3] http://www.oracle.com/technology/documentation/database10gr2.html
[4] http://www.bookpool.com/sm/0072230533
[5] http://www.redhat.com/docs/manuals/csgfs/Oracle_GFS-en-US/index.html
[6] http://www.redhat.com/docs/manuals/csgfs/pdf/Oracle_GFS.pdf
[7] http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/release-notes/as-x86/
[8] http://www.oracle.com/technology/pub/notes/technote_rhel3.html
[9] http://www.redhat.com/whitepapers/rhel/AdvServerRASMpdfRev2.pdf
[10] http://www.redhat.com/whitepapers/rhel/OracleonLinux.pdf
[11] http://www.redhat.com/whitepapers/rhel/OracleonLinux.pdf
[12] http://www.redhat.com/magazine/001nov04/features/vm/
[13] http://www.bookpool.com/sm/0596005652
[14] http://kerneltrap.org/node/2450
[15] http://www.oracle.com/technology/deploy/availability/pdf/S939_SusairajLee.ppt.pdf

Oracle MetaLink Note:249213.1

Oracle MetaLink Note:260152.1

Oracle MetaLink Note:262004.1

Oracle MetaLink Note:265194.1

Oracle MetaLink Note:270382.1

Oracle MetaLink Note:280463.1

Oracle MetaLink Note:329378.1

Oracle MetaLink Note:344320.1

Oracle Database Documentation Library[16]

Oracle Validated Configurations[17]

[16] http://www.oracle.com/pls/db102/portal.portal_demo3?selected=1
[17] http://www.oracle.com/technology/tech/linux/validated-configurations/index.html

Appendix A. Revision History

Revision 1.00 Chris Curran *ccurran@redhat.com*

Converted to the Red Hat standard

Revision 1.01 Chris Curran *ccurran@redhat.com*

Legal Notice Page: Clarification of Copyright Statement. Red Hat owns the copyright to this and all other subsequent derivative works. Werner Puschitz retains the copyright to the original at www.puschitz.com.
Chapter 2, 32 bit Architecture: Added “In Red Hat Enterprise Linux 5, a 32 bit kernel is always a hugemem kernel so there is no need to install a special kernel.”
Chapter 4, The I/O Scheduler: Added “Red Hat Enterprise Linux 5 in fact allows users to change I/O schedulers dynamically (ie echo sched_name > /sys/block/<sdx>/queue/scheduler)”
Chapter 5, Tuning the Page Cache: Added “For Red Hat Enterprise Linux4/5, the pagecache is dynamically adjusted. You can adjust the minimum free pages using; # echo 1024 > /proc/sys/vm/min_free_kbytes Again to make the change permanent,add the following line to the file /etc/sysctl.conf; # echo vm.min_free_kbytes=1024 >> /etc/sysctl.conf Additional tuning can be done to start reclaiming pagecache pages by adjusting the swappiness percentage as described in the next section.”
Chapter 12: Changed title to “Enabling Asynchronous I/O and Direct I/O Support
Chapter 12, Enabling Asynchronous I/O in Oracle 9i and 10g: Added “If you use file systems instead of raw devices or ASM for data files, then you need to ensure that the datafiles reside on file systems that support asynchronous I/O (e.g., OCFS/OCFS2, ext2, ext3). To do async I/O on file systems the filesystemio_options parameter needs to be set to "asynch". Eliminated all references to disk_asynch_io=true Added “For Red Hat Enterprise Linux 3, it is recommended you use direct I/O ONLY for ext2, ext3, GFS, NFS and OCFS file systems. For Red Hat Enterprise Linux 4/5, it is strongly recommended to “setall” for ext2, ext3, GFS, NFS and OCFS file systems.”
Chapter 14: Added Note “As explained in detail in this section, enabling big pages helps reduce TLB misses. However, this performance benefit is realized primarily when using large SGA sizes. Once a portion of memory is locked down for big pages, applications that use normal pages cannot access that portion of the memory. It is very important to make sure that there is enough memory for normal pages for applications and users to avoid excessive swapping. So, it is recommended that big pages be used only on systems that have large amounts of physical memory and for SGA sizes of 16GB or greater.”
Chapter 18: Deleted chapter on Oracle's Orion performance prediction tool

Revision 1.1 November 2007 Chris Curran *ccurran@redhat.com*

Added Part II - Installing the Oracle Database 10g on Red Hat Enterprise Linux
Added Part III - Installing the Oracle9i 32 bit Database on Red Hat Enterprise Linux

Revision 1.4 September 2008 Chris Curran *ccurran@redhat.com*

Updated to the latest version of publican and pushed to brew.